HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

DIPLOMARBEIT

Herr
Rocco Oehme

Spezifikation, prototypische Implementierung und Test einer Applikation zur GPU-gestützten Bildverarbeitung und Texterkennung am Beispiel des mobilen Betriebssystems Apple iOS

Mittweida, 2016

Fakultät Medien

DIPLOMARBEIT

Spezifikation, prototypische Implementierung und Test einer Applikation zur GPU-gestützten Bildverarbeitung und Texterkennung am Beispiel des mobilen Betriebssystems Apple iOS

Autor:
Herr

Rocco Oehme

Studiengang:
Multimediatechnik

Seminargruppe:
MK11w1-D

Erstprüfer:
Prof. Dr.-Ing. Alexander Lampe

Zweitprüfer:
Dipl.-Ing. (FH), M. Sc. Rico Thomanek

Einreichung:
Mittweida, 05.02.2016

Verteidigung/Bewertung:
Mittweida, 2016

Faculty Media

DIPLOMA THESIS

Specification, prototype implementation and testing of an application for GPU-based image processing and text recognition using Apple`s mobile operating system iOS

author:
Mr.

Rocco Oehme

course of studies:
<bezeichn d. studienganges>

seminar group:
MK11w1-D

first examiner:
Prof. Dr.-Ing. Alexander Lampe

second examiner:
Dipl.-Ing. (FH), M. Sc. Rico Thomanek

submission:
Mittweida, 05.02.2016

defence/ evaluation:
Mittweida, 2016

Bibliografische Beschreibung:

Oehme, Rocco:
Spezifikation, prototypische Implementierung und Test einer Applikation zur GPU-gestützten Bildverarbeitung und Texterkennung am Beispiel des mobilen Betriebssystems Apple iOS. – 2016. – IX, 51, 35 S.
Mittweida, Hochschule Mittweida, Fakultät Medien, Diplomarbeit, 2016

Referat:

Diese Diplomarbeit beschreibt die Entwicklung und Umsetzung einer Applikation zur Bewertung von Banknoten verschiedener Währungen für das Apple iPad. Dabei wird neben dem Gesamtsystem auch der Ablauf der Software vorgestellt und die Hauptfunktionen des Votingvorganges an Quelltextauszügen kurz erläutert. Zum Abschluss der Arbeit wird ein Ausblick gegeben, welche Erweiterungen am Projekt sinnvoll und möglich wären.
Im Anhang dieser Diplomarbeit wurden kurze Benutzerhandbücher zu den einzelnen Funktionen der App und des Webservers angefügt.

Inhalt

Abbildungsverzeichnis

Abkürzungsverzeichnis

Ad-hoc Netz	Funknetz, welches zwei oder mehr Endgeräte zu einem vermaschten Netz verbindet.
App	Anwendungssoftware für Mobilgeräte
ASCII	„American Standard Code for Information Interchange“, ist eine 7-Bit-Zeichenkodierung
C++	Erweiterung der Programmiersprache C
CoreData	Framework zum Verwalten der Model Layer Objekte eines Programms
CPU	„Central Processing Unit“, Prozessor eines Computers
float	Datentyp für Gleitkommazahlen
GPU	„Graphics Processing Unit“, Prozessor zur Berechnung von Grafiken
GPUImage	iOS Framework für GPU basierte Bild- und Videoverarbeitung
iOS	von Apple entwickeltes Betriebssystem für mobile Endgeräte
IP	Adresse in Computernetzen
iPad	Tablet-Computer des Herstellers Apple
JSON	„JavaScript Object Notation“, kompaktes Datenformat zum Zweck des Datenaustauschs zwischen Anwendungen
LED	„Light-Emitting Diode“ (Leuchtdiode)
Mac OS	von Apple entwickeltes Betriebssystem für Macintosh-Computer
MySQL	relationales Datenbankverwaltungssystem
Objective C	Erweiterung der Programmiersprache C um Sprachmittel zur objektorientierten Programmierung
OCR	„Optical Character Recognition“, automatisierte Texterkennung innerhalb von Bildern
Open GL	„Open Graphics Library“, Spezifikation für eine plattform- und programmiersprachenunabhängige Programmierschnittstelle zur Entwicklung von 2D- und 3D- Computergrafikanwendungen
PC	„Personal Computer“
PHP	Scriptsprache zur Erstellung dynamischer Webseiten oder Webanwendungen

PIN	„Persönliche Identifikationsnummer“ zum authentisieren gegenüber einer Maschine
RGB	„Rot, Grün und Blau“ Werte im RGB-Farbraum
SDL	„Spezifikations- und Beschreibungssprache“ um Systeme mittels erweiterter Zustandsautomaten zu beschreiben
UInt8	„Unsigned Integer 8“, Datentyp Integer mit 8 Bit und ohne Vorzeichen
WEP	„Wired Equivalent Privacy“, ist ein Verschlüsselungsprotokoll für WLAN
WLAN	„Wireless Local Area Network“, ist ein lokales Funknetz
XAMPP	ist eine Zusammenfassung freier Software; ermöglicht Installation und Konfiguration eines Webservers mit einer Datenbank
XCode	Entwicklungsumgebung von Apple zur Programmentwicklung für Apple eigene Betriebssysteme

1 Einleitung

Die Programmierung von Applikationen für mobile Endgeräte hat sich in den letzten Jahren zu einem essenziellen Pfeiler in der Informatik entwickelt. Allein in Deutschland benutzen (Stand Juli 2015) ca. 46 Millionen Menschen ein Smartphone oder Tablet-PC. Damit verwenden mehr als die Hälfte aller Deutschen ein mobiles Endgerät. Daran verdeutlicht sich wie groß die Nachfrage an guten Applikationen in diesem Bereich ist.

Allerdings bezieht sich dies nicht nur auf den privaten Bereich. Eine Großzahl deutscher Unternehmen verwendet die kleinen praktischen Helfer auch im Arbeitsalltag. Dies ist aber nicht bei allen Firmen ohne weiteres möglich, da spezielle Aufgaben nur mit speziellen Applikationen erfüllt werden können.

1.1 Motivation und Zielstellung

Eine „App“ ist im deutschen Sprachgebrauch und nach den meisten Definitionen eine Anwendungssoftware für Mobilgeräte bzw. mobile Betriebssysteme. Der private Nutzer stellt sich darunter im Allgemeinen ein Programm zur Information oder Unterhaltung vor. Das Applikationen auf mobilen Endgeräten auch einen praktischen Nutzen haben und zur Lösung verschiedener Problemstellungen führen können, wissen leider die Meisten nicht.

Viele Unternehmen nutzen Apps um ihren Arbeitsalltag zu erleichtern oder um Ihre Produktpalette ausbauen zu können. Dies wird in den häufigsten Fällen nur durch, an die Anforderung angepasste Programme realisiert. Es ist aber sehr selten, dass solche Apps, welche die vorhandenen Aufgaben lösen können, zur freien Verfügung stehen. Daher lassen die hochmobilen Unternehmen eigens für die vorhandenen Problemstellungen Applikationen entwickeln und umsetzen.

Mit genau dieser Vorgehensweise möchte sich die CI Tech Components AG auf dem Gebiet der Fitnessbewertung von Banknoten vergrößern. Man sucht nach einer Softwarelösung für das Apple iPad, bei der der Zustand verschiedener Banknoten mit Hilfe einer App bewertet werden kann. Dies soll auf visueller Inspektion und Eingabe des Ergebnisses basieren.

Die wesentlichen Anforderungen der CI Tech Components AG bestehen aus folgenden Punkten. Zum einen soll die grafische Benutzeroberfläche intuitiv sein, damit der Bewertungsablauf auch Spaß macht (Smart-Faktor), zum anderen soll der Testablauf auch zuverlässig und einfach sein. Eine weitere wichtige Anforderung ist die verlässlich funktionierende, automatische Seriennummernbestimmung (OCR).

Der spezielle Funktionsablauf und -umfang für das Voting beinhaltet das Erstellen mehrerer Testdecks. Dabei soll der Name des Testdecks nach einer Vorgabe (Währung_Denomination_Emission_Jahr_(AnzahlBanknoten)_Kommentar) angelegt werden. Ein solches Testdeck kann sowohl aus fitten wie auch aus unfitten Banknoten bestehen und beinhaltet maximal 100 Geldscheine. Die Seriennummern der Noten werden mit einem externen Programm erfasst und in einer Mutterliste abgelegt. Diese Mutterliste wird für den Votingprozess im Tablet zwischengespeichert und als Vergleichsparameter für die Texterkennung verwendet. Wenn keine Liste zum jeweiligen Testdeck vorhanden ist, soll das Voting trotzdem durchführbar sein und eine Mutterliste zu den ermittelten Ergebnissen angelegt werden.

Eine weitere Funktion der Applikation ist das Login des Probanden. Dies soll mittels Vorname und Name durchführbar sein. Außerdem muss eine Neuanmeldung eines bisher nicht angelegten Nutzers möglich sein. Der Benutzer soll nach erfolgreicher Anmeldung ein Testdeck auswählen können, welches vorher bereits randomisiert wurde. Daraufhin erstellt das Tablet eine Tabelle mit dem Namen der Mutterliste in welcher die Daten gespeichert werden. Nach Abschluss dieser Funktionen startet das Voting zum Prozessieren aller Banknoten des ausgewählten Testdecks. Dabei soll folgender Ablauf befolgt werden.

Der Proband platziert die erste Banknote des Testdecks im Anschlag unter der Kamera des Tablets so, dass die Seriennummer sichtbar ist. Per Button soll ein Foto erstellt werden, welches als Vorlage für die OCR dient. Daraufhin soll die Texterkennung starten und mittels Rahmen die ermittelten Bereiche ausgeben, in denen die Seriennummern liegen und von denen der Text ausgelesen wird. Wenn eine Mutterliste vorhanden ist, müssen die Ergebnisse der OCR mit den Werten der Liste verglichen werden. Die Ergebnisse der Texterkennung werden dann ausgegeben und können vom Benutzer bearbeitet werden, oder bei Übereinstimmung gespeichert werden.

Daraufhin folgt das Bewerten der Banknoten, welches in zwei Modi aufgeteilt ist. Der Benutzer kann zwischen dem Modus „Fit“ und dem Modus „Unfit“ wählen. Wenn die Banknote dem Modus „Fit“ entspricht, wird dies gespeichert. Bei einer unfitten Note soll der Proband eine Mehrfachnennung mit den jeweiligen Gründen für den Zustand treffen können. Nach dem Bewerten des Geldscheines ist das Voting abgeschlossen und kann mit der nächsten Banknote fortgesetzt werden oder per Knopfdruck beendet werden.

Die gespeicherten Daten auf dem Tablet sollen vom Experten einfach zum Laptop für die Weiterverarbeitung transferiert werden können bzw. sollen die Mutterlisten vom Laptop zum Tablet-PC kopiert werden können.

1.2 Aufbau der Arbeit

Die vorliegende Diplomarbeit wird in folgende Kapitel unterteilt:

Kapitel 1 erläutert das Thema der Diplomarbeit und beschreibt die Motivation und Zielstellung des Projektes, welches großer Bestandteil der Arbeit ist. Außerdem wird eine Kapitelübersicht gegeben.

Kapitel 2 bildet das komplette System des Projektes ab und erläutert den Aufbau von erzeugten Testdecks. Des Weiteren zeigt das Kapitel, welche Anforderungen an die Applikation gestellt werden und mit welchen Hilfsmitteln diese umgesetzt werden können. Diese Hilfsmittel werden kurz beschrieben.

Kapitel 3 beschreibt den Entwurf der Vorrichtung, welche zum Bewerten der Banknoten benötigt wird. Ebenso wird der Ablauf der Applikation mittels der Ablaufdiagramme der Automaten beschrieben.

Kapitel 4 beinhaltet die Implementierung der FitnessVoter-App, welche der Hauptbestandteil der Diplomarbeit ist. Dabei werden die Hauptfunktionen der Applikation beschrieben und mittels Grafiken verdeutlicht.

Kapitel 5 gibt einen Rückblick über den gesamten Aufwand der Diplomarbeit. Darin werden die gewonnenen Erkenntnisse und Erfahrungen, welche während des Bearbeitungszeitraums gesammelt wurden, dargestellt.

Kapitel 6 bildet den Abschluss dieser Diplomarbeit und zeigt die Erweiterungs- und Verbesserungsmöglichkeiten des Gesamtsystems, basierend auf eigenen Erfahrungen und Anmerkungen des Auftraggebers CI Tech Components AG.

2 Anforderungen an die Applikation

Um die Applikation funktionstüchtig umsetzen zu können, mussten in der Entwicklungsphase des Projektes einige Entscheidungen getroffen werden. Dabei wurden die Anforderungen der CI Tech Components AG berücksichtigt und die bestmöglichen Komponenten ausgewählt. Die wichtigsten Entscheidungen werden in folgenden Abschnitten nochmals genauer beschreiben.

2.1 Anforderungsanalyse

Bevor jedoch spezielle Anforderungen an das System gestellt werden können, müssen die einzelnen Komponenten in ein Gesamtpacket zusammengefasst werden. Dabei sollte der prinzipielle Ablauf beachtet werden. Um dies gut zu verdeutlichen, wurde ein Systemabbild erzeugt.

2.1.1 Systemabbild

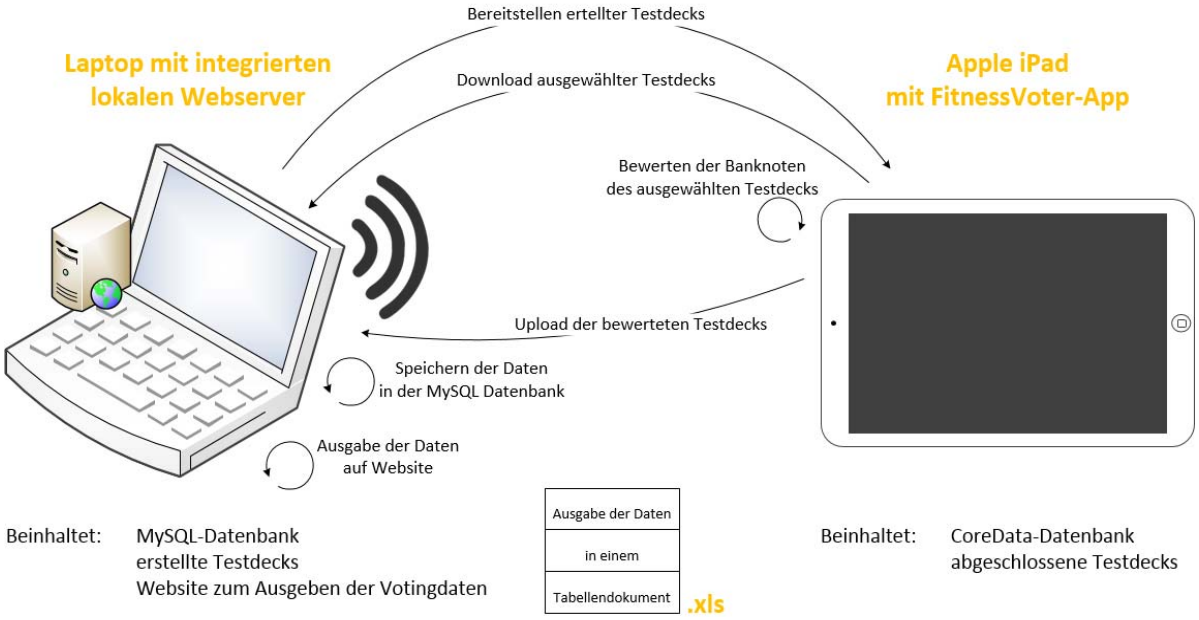


Abbildung 1

Das gesamte System besteht aus einem Notebook mit Windows Betriebssystem und einem Apple iPad Air 2. Diese beiden Geräte sollen drahtlos miteinander kommunizieren und dabei Daten austauschen können. Jedoch soll keine ständige Verbindung zwischen beiden Geräten bestehen. Die Funktionen des einzelnen Gerätes sollen auch ohne Verbindung ausführbar sein.

Der Windows Laptop besitzt einen lokalen Webserver mit integrierter Datenbank, worüber eine Kommunikation mit dem Tablet-PC möglich ist. Auf diesem Webserver soll ein PHP-Formular zur Erstellung eines Testdecks implementiert sein. Nach Abschicken dieses Formulars, müssen die erzeugten Daten in der Datenbank gespeichert werden und für einen Datentransfer vorbereitet werden.

Das Apple iPad schickt eine Anfrage an den Webserver des Notebooks, um auf die angelegten Testdecks zugreifen zu können. Nach einer Auswahl durch den User, wird das jeweilige Testdeck vom Server heruntergeladen und auf dem Tablet-PC in einer Datenbank gespeichert. Dort können diese Daten zum Bewerten von Banknoten verwendet werden.

Ist der Votingvorgang beendet und das Testdeck abgeschlossen, kann das Testdeck an den Server zurückgesendet werden. Der Server bekommt die Daten der Bewertung und speichert diese in einer Tabelle der Datenbank. Letztendlich sollen diese Daten, mittels einer PHP-Datei, im Browser des Windows-Notebooks ausgegeben werden. Außerdem soll die Möglichkeit bestehen, die Daten in ein Tabellendokument zu schreiben, um diese im weiteren Verlauf Auswerten zu können.

2.1.2 Aufbau eines Testdecks

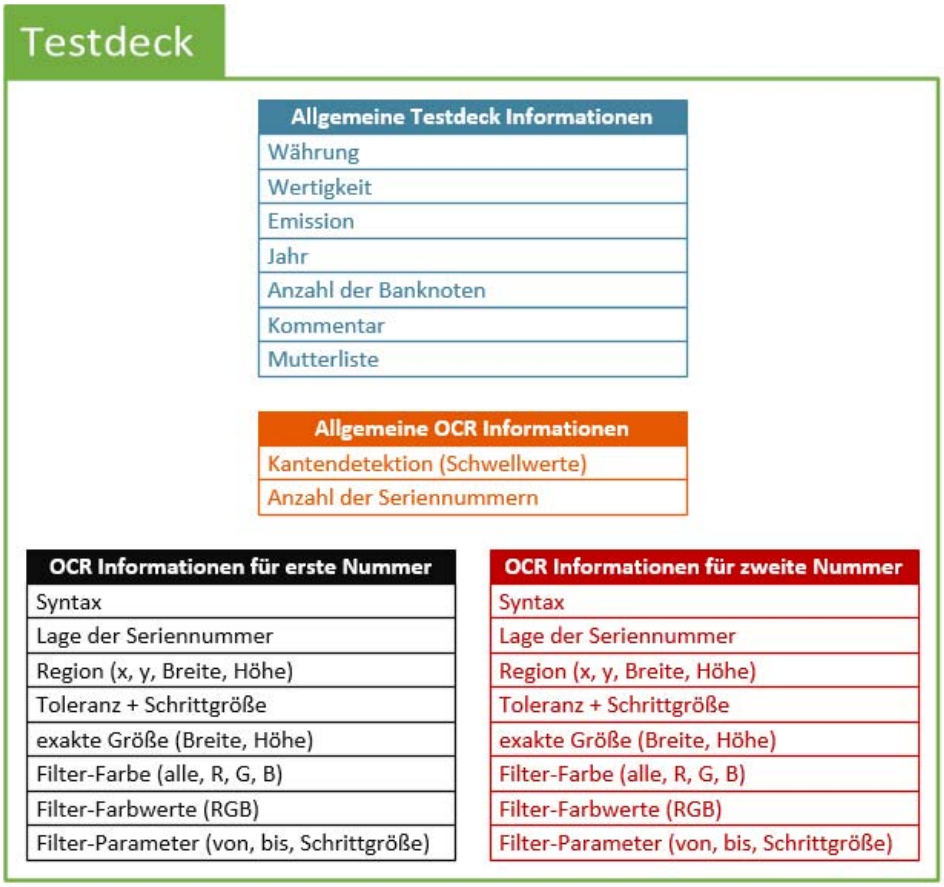


Abbildung 2

Ein am Webserver erzeugtes Testdeck ist immer gleich aufgebaut, damit die Applikation auf dem Apple iPad stets mit den Parametern umgehen kann. Jedes Testdeck beginnt mit allgemeinen Angaben. Diese beinhalten die Währung, die Wertigkeit und die Emission der Banknote. Außerdem wird die Anzahl der Geldscheine festgelegt und, wenn vorhanden, eine Mutterliste zum jeweiligen Geldbündel mitgeliefert.

Als nächstes folgen allgemeine Informationen für die OCR. Dabei werden zur Ermittlung der Kanten der Banknote drei Schwellwerte (Kanten-Stärke, Kanten-Schwellwert und Linien-Ermittlungs-Schwellwert) angelegt. Des Weiteren wird angegeben, wie häufig die Seriennummer auf der Banknote zu finden ist.

Nach den allgemeinen Informationen folgen spezielle Angaben für die jeweilige Seriennummer. Diese Angaben sind essenziell für die Texterkennung. Als Erstes wird die Syntax der Seriennummer angegeben, welche später als Vergleichswert genutzt werden soll. Hierbei wird ebenfalls eine Liste aller möglichen Zeichen, die in der Seriennummer vorkommen dürfen, angegeben und es müssen die Positionen des jeweiligen Zeichens der Syntax beschrieben werden, da später die OCR-Ergebnisse Zeichen für Zeichen mit anderen Ergebnissen verglichen werden. Außerdem wird die Lage (horizontal oder vertikal) der Seriennummer festgelegt. Es wird auch eine Region des Geldscheines mitgeliefert, in der die Seriennummer zu finden ist. Dazu werden vier Werte (x, y, Breite, Höhe) in das Testdeck eingetragen, wobei die Koordinaten von der oberen linken Ecke der Banknote zu betrachten sind.

Gleichzeitig gibt der Erzeuger des Testdecks eine Toleranz zu der Region an, um mittels der Schrittgröße auch die umliegenden Bereiche für die OCR bereitzustellen. Diese werden allerdings nur betrachtet, wenn keine korrekte Seriennummer in der Region gefunden wurde. Ein weiterer Vergleichswert, ist die exakte Größe der Nummer, welche ebenfalls mit Breite und Höhe angegeben wird. Um das Bild für die Texterkennung so gut wie möglich zu filtern, wird als erstes ein Wert (R, G, B oder alle Farben) für die Filterfarbe angegeben. Anschließend können für die jeweilige Farbe spezielle Zahlenwerte gesetzt werden. Daraufhin werden Filterparameter für den Average-Luminance-Filter angegeben. Diese werden mittels eines Minimal- und Maximalwertes und einer Schrittgröße erzeugt.

Besitzt die Banknote, des zu bewertenden Testdecks, nur eine Seriennummer, werden auch nur Werte zu dieser einen Nummer angegeben. Sind jedoch zwei Nummern auf dem Geldschein zu finden, so werden diese speziellen Angaben für jede Seriennummer angegeben.

2.2 Gerät- und Programmiersprachenauswahl

Die Entscheidung eine Applikation für das Apple iPad zu entwickeln war relativ klar. Der Vorteil darin liegt an der Nachhaltigkeit. Da bei einer Umsetzung mit Android viele Produkthersteller mit verschiedenen Gerätetypen und unterschiedlichsten Bildschirmauflösungen vorhanden sind, ist die optimale Anpassung der Software an das jeweilige Gerät ziemlich schwierig. Wo hingegen bei der Umsetzung mit iOS lediglich das Apple iPad zur Verfügung steht, welches mit einer gleichbleibenden Bildschirmauflösung, auch bei den Nachfolgermodellen, punkten kann. Somit ist die Verwendung des Programms auch für die nahe Zukunft gesichert.

Die Auswahl eines mobilen Endgerätes entscheidet aber auch gleichzeitig über die Verwendung der Programmiersprache. So hat man bei der Entwicklung mit iOS die Wahl zwischen Objective C und Swift. Die Programmiersprache Swift ist eine noch sehr neue Programmiersprache, welche noch in der Entwicklungsphase steht und noch nicht ganz ausgeklügelt ist. Daher hat man sich für eine Umsetzung mit der Programmiersprache Objective C entschieden.

Objective C erweitert die Programmiersprache C um Sprachmittel zur objektorientierten Programmierung. Ein wichtiger Gedanke bei der Entwicklung von Objective C war es, sich der Flexibilität von Smalltalk anzunähern. Dabei soll auf alles verzichtet werden, was das Laufzeitverhalten verschlechtert.

Ein großer Vorteil von Objective C gegenüber Swift ist, dass unzählige Frameworks frei zur Verfügung stehen. Somit kann man ein Projekt in die verschiedensten Richtungen erweitern. Weitere Vorteile sind unter anderem das Senden von Nachrichten innerhalb eines Programms oder das dynamische Binden von Methoden.

2.3 Kantendetektion

Ein ganz wichtiger Bestandteil im Ablauf der Software ist die Ermittlung der Kanten der Banknote, da diese nur ein Teil des Bildes ist. Die Kantendetektion wird zum einen für das Auslesen der Seriennummer des Geldscheines benötigt und ist zum anderen essenziell für das Vorbereiten des Bildausschnittes, welcher für den Upload an den Webserver benötigt wird. Für die Umsetzung der Kantenermittlung wird die Schwellwertbestimmung benötigt.

Die Schwellwertbestimmung ist notwendig, um die Kanten des Geldscheins auf dem Bild für die Ermittlung greifbar zu machen und ist somit ein wichtiger Teil für die Bildanalyse. Dabei wird mittels des Schwellenwertverfahrens ein Binärbild erzeugt.

Darunter versteht man den Vergleich des Grauwertes oder eines anderen eindimensionalen Merkmales eines Pixels mit einem Schwellenwert. Bei der Überschreitung dieses Schwellenwertes erhält das Pixel den Wert 1, ansonsten erhält es den Wert 0. Somit ent-

steht nach dem Schwellenwertverfahren ein Schwarz-Weiß Bild. Anhand dieses Bildes werden die Kanten der Banknote verdeutlicht und können durch pixelweißes Abschreiten von allen vier Seiten ermittelt werden.

2.4 Bildfilterung

Um das Bild für die OCR optimal vorzubereiten ist es notwendig mittels eines Filters die Seriennummer der Banknote deutlich darzustellen, ohne Überlagerung der Hintergrundgrafik des Geldscheines. Da bei einigen Währungen auch die Seriennummer farbig auf der Banknote gedruckt ist, werden verschiedene Farbfilter eingesetzt. Für die Umsetzung dieser Filter wird das GPUImage – Framework verwendet.

GPUImage ist eine freie iOS-Bibliothek die es erlaubt, GPU bezogene Filter und Effekte auf Bilder, Filme und dem Live-Kameravideo anzuwenden. Diese Bibliothek liefert eine Großzahl an Filter, die es ermöglichen die Banknote optimal für die Texterkennung vorzubereiten. Außerdem nutzt GPUImage Open GL ES 2.0 shaders, die eine deutlich schnellere Bearbeitung der Bilder und Videos gewährleisten, als es mit der CPU möglich wäre.

Auch das Integrieren des Frameworks in ein bestehendes XCode-Projekt ist in wenigen Schritten realisiert. Die GPUImage Bibliothek ermöglicht eine deutliche Zeitersparnis beim Implementieren der Foto-Filter, da diese mit einem einfachen Methodenaufruf benutzt werden können.

2.5 Texterkennung

Bei jeder Fitnessbewertung eines Geldscheines soll die Seriennummer erfasst werden. Da lediglich ein Bild zur Verfügung steht, auf dem diese Nummer vorhanden ist, kann das Auslesen nur durch eine OCR realisiert werden.

Diese optische Zeichenerkennung funktioniert allerdings nur fehlerfrei, wenn der auszulesende Text vorher optimal vorbereitet wurde. Des Weiteren muss der Texterkennungssoftware der benötigte Schriftfont angelernt werden, um einen Vergleichswert für die ermittelten Zeichen zu besitzen.

Tesseract ist eine freie Software zur optischen Zeichenerkennung mit guten Ergebnissen auf der Zeichenebene, welche auch unter Mac OS verfügbar ist. Es wurde in der Programmiersprache C++ entwickelt und ist somit problemlos unter Objective C nutzbar.

Das Anlernen neuer Schriftfonts ist mit Hilfe des jTessBoxEditors auch sehr übersichtlich durchführbar und gut nachzuvollziehen. Tesseract liefert eine Menge an Funktionen, die eine spezielle Anpassung der Texterkennung bieten. Die Integration der OCR wird mittels Pods umgesetzt.

2.6 Datenspeicherung auf mobilem Endgerät

Da eine ständige Verbindung zum lokalen Webserver nicht gewährleistet werden kann, müssen die Daten der Banknotenbewertung vorerst auf dem mobilen Endgerät zwischengespeichert werden. Das ermöglicht, bei Verbindung zum Server, einen Upload mehrerer Datensätze, welcher später durch den Administrator gesteuert werden kann.

Diese Speicherung der bewerteten Geldscheine mit den dazugehörigen Daten wird mittels CoreData umgesetzt. Das CoreData Framework ist ebenfalls eine freie Bibliothek zur Speicherung von Daten. Es ist allerdings kein Datenbanksystem, aber es verwaltet die Datenobjekte gemäß eines grafisch editierbaren relationalen Datenmodells. CoreData unterstützt dabei bidirektionale 1:n und n:m Beziehungen zwischen Objekten.

Die Daten werden als Attribute in Entities gespeichert und können über diese Zuordnung auch wieder aufgerufen werden. Der Vorteil dieser Speicherung ist der Zugriff mittels Methoden auf Daten in jeder Klasse des Projektes. Somit müssen bestimmte Werte nicht mehr an jede Klasse übergeben werden, in der sie benötigt werden. Auch dieses Framework lässt sich einfach und in wenigen Schritten zum bestehenden XCode-Projekt hinzufügen.

2.7 Verbindung zwischen iPad und PC

Nach der Bewertung der Banknoten sollen die Daten an einen Windows PC übermittelt werden. Da dies mit einem Apple iPad allerdings nicht so einfach möglich ist, musste eine elegante Lösung gefunden werden. Bei dem Windows PC handelt es sich um ein Notebook, welches einen drahtlosen Netzwerkadapter besitzt. Somit ist es möglich die beiden Geräte über eine drahtlose Verbindung kommunizieren zu lassen.

Um diese Verbindung umzusetzen, gibt es zwei Möglichkeiten. Die erste Möglichkeit wäre, die beiden Geräte mittels eines Routers in einem Netzwerk zu verbinden. Allerdings benötigt man dafür ein Zusatzgerät (Router), welches nicht unbedingt nötig ist. Die zweite Möglichkeit ist, die direkte Verbindung der beiden Geräte über eine drahtlose Verbindung, welche man über ein Ad-hoc-Netz realisiert.

Unter einem Ad-hoc-Netz versteht man in der Informatik ein Funknetz, das zwei oder mehr Geräte zu einem vermaschten Netz verbindet. Diese Netze verbinden mobile Geräte ohne feste Infrastruktur wie Wireless Access Points.

Um den Datentransfer zwischen dem Apple iPad und dem Windows Notebook herstellen zu können, wird bei der Verbindung der beiden Geräte ein Ad-hoc-Netz vom drahtlosen Netzwerkadapter des Notebooks ausgestrahlt. Dieses Netz ist mittels WEP-Verschlüsselung gesichert.

Das Einrichten einer solchen Ad-hoc-Verbindung ist in wenigen Schritten umsetzbar und bleibt auch als gespeichertes Netzwerk am PC erhalten. In dieses Netz wählt sich das Tablet mit dem vergebenen Netzwerkschlüssel ein und hat somit eine direkte Verbindung zum Windows PC hergestellt. Über dieses Netz können nun die Daten zwischen den beiden Geräten ausgetauscht werden. Somit kann man komplett auf die Verwendung eines externen Gerätes, wie zum Beispiel eines Wireless Access Points, verzichten.

2.8 Transport der Daten

Nach dem Aufbau einer Verbindung zwischen dem Apple iPad und dem Windows Notebook, ist der Austausch der Daten eine weitere wichtige Aufgabe, die realisiert werden musste. Um dies umzusetzen, wurde nach einem Format gesucht, welches das Senden der Daten erlaubt und von beiden Geräten verarbeitet werden kann.

Eine freie und Programmiersprachen unabhängige Variante ist JSON, welche auch im Projekt „FitnessVoter“ verwendet wird.

Die JavaScript Object Notation (JSON) ist ein kompaktes Datenformat zum Zweck des Datenaustauschs zwischen Anwendungen. Da in praktisch allen verbreiteten Sprachen Parser existieren, ist es für die Kommunikation eines Mac-basierten und eines Windows-basierten Gerätes geeignet. Somit ist die Kommunikation zwischen dem Apple iPad und dem Windows Notebook gewährleistet und es können Daten empfangen und gesendet werden.

2.9 Verarbeitung & Speicherung der Daten auf dem PC

Eine weitere Anforderung an das System, ist die Ausgabe der empfangenen Daten auf einer Webseite. Um diese Anforderung umzusetzen ist ein Webserver notwendig der lokal auf dem Windows PC installiert ist. Dafür bietet sich das XAMPP-Paket an, da es eine Zusammenstellung freier Software ist. Für die Ausgabe der Daten ist es wichtig, dass sie vorher auf dem Notebook gespeichert werden. Diese Aufgabe übernimmt auch das XAMPP-Paket, da es eine Datenbank beinhaltet.

XAMPP ermöglicht die einfache Installation und Konfiguration des Webservers Apache mit einer MySQL-Datenbank und den Skriptsprachen Perl und PHP. MySQL ist ein weit verbreitetes Datenbankverwaltungssystem, welches die Grundlage für viele dynamische Webauftritte bildet.

Um die vom iPad gesendeten Daten verarbeiten zu können, werden diese mit Hilfe einer PHP-Datei ausgewertet und in die MySQL-Datenbank des Webservers geschrieben und gespeichert. Eine weitere PHP-Datei greift auf diese Daten zu und gibt diese in einer Tabelle auf einer Webseite aus. Dies ist nur mittels MySQL-Befehle möglich.

3 Entwurf

Vor der Umsetzung des Projektes „FitnessVoter“ mussten die Anforderungen der CI Tech Components AG greifbar gemacht und in einen klaren Ablauf übertragen werden. Um den Ablauf der Applikation zu verdeutlichen, wurden SDL-Diagramme erarbeitet, welche die Funktionen der Software Schritt für Schritt abbilden. Somit war eine Schrittweise Programmierung der Applikation möglich.

Außerdem musste die Grundidee des Aufbaus der Voting-Vorrichtung geprüft und skizziert werden, damit diese auch in die Praxis umgesetzt werden konnte. Dafür mussten auch ein paar kleine Änderungen vorgenommen werden, im Vergleich zur ersten Überlegung.

3.1 Grundaufbau der Voting-Vorrichtung

Um alle Funktionen der Applikation bestmöglich nutzen zu können, ist eine Vorrichtung für das Voting unabdingbar. Da es bei der Texterkennung wichtig ist die Banknote bestmöglich vorzubereiten, ist ein gleichbleibender Abstand zur Kamera und eine gleichbleibende Position der Banknote sehr vorteilhaft. Außerdem richtet sich die Aufmerksamkeit des Benutzers mehr auf die Banknote, als auf das Erzeugen eines Bildes der Note in einer guten Qualität, wodurch eine intensivere Bewertung des Geldscheines möglich ist.

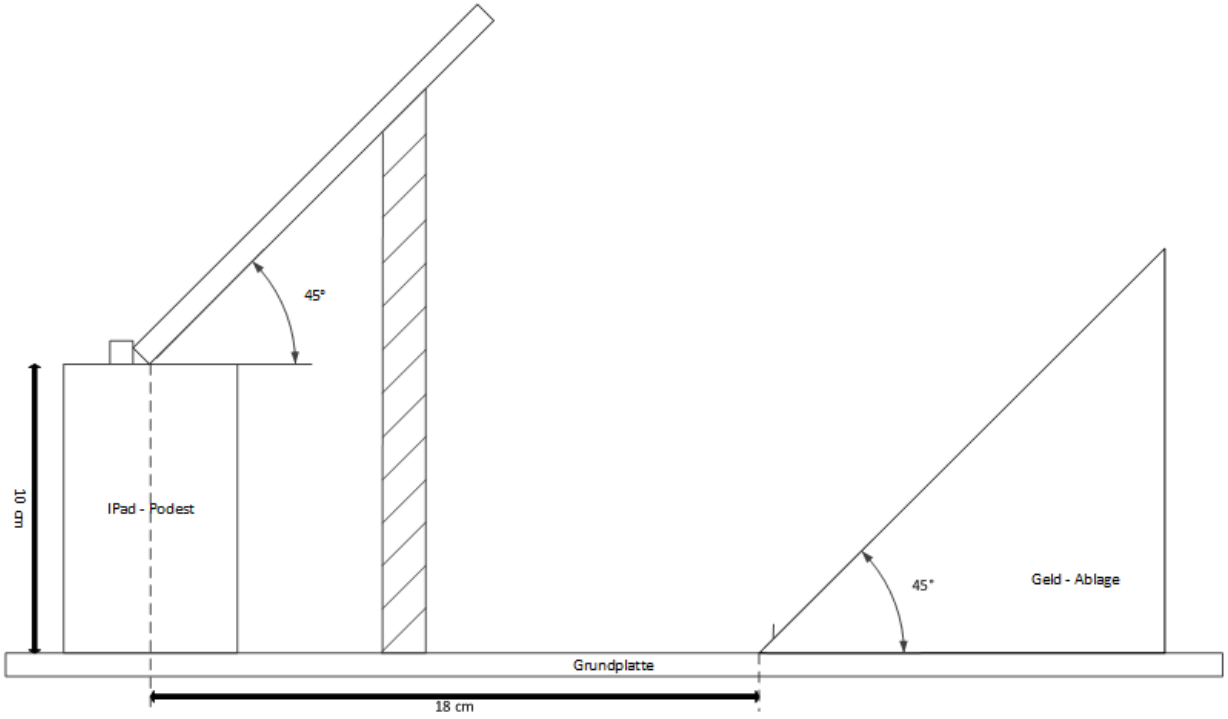


Abbildung 3

Bei der Entwicklung des Grundaufbaus für die Vorrichtung zum Bewerten von Banknoten war ein essenzieller Punkt die Bedienung nutzerfreundlich zu gestalten. Daher wurde das Tablet in einem 45° Winkel in 10 Zentimeter Höhe angebracht. Dies lässt eine gute Bedienung aus verschiedenen Positionen (stehend oder sitzend) zu. Ein weiterer Punkt dafür ist die Geldablage, welche mit einer Entfernung von 18 Zentimetern zum Tablet-PC leicht erreichbar ist. Diese Ablage ist ebenfalls in einem 45° Winkel angebracht und mit einer Leiste an der Unterkante versehen, welche das Abrutschen der Banknoten verhindert. In diesem Winkel liegt der Geldschein auch so, um gut erkennbar zu sein und um nicht nach vorn zu kippen und von der Ablage zu fallen.

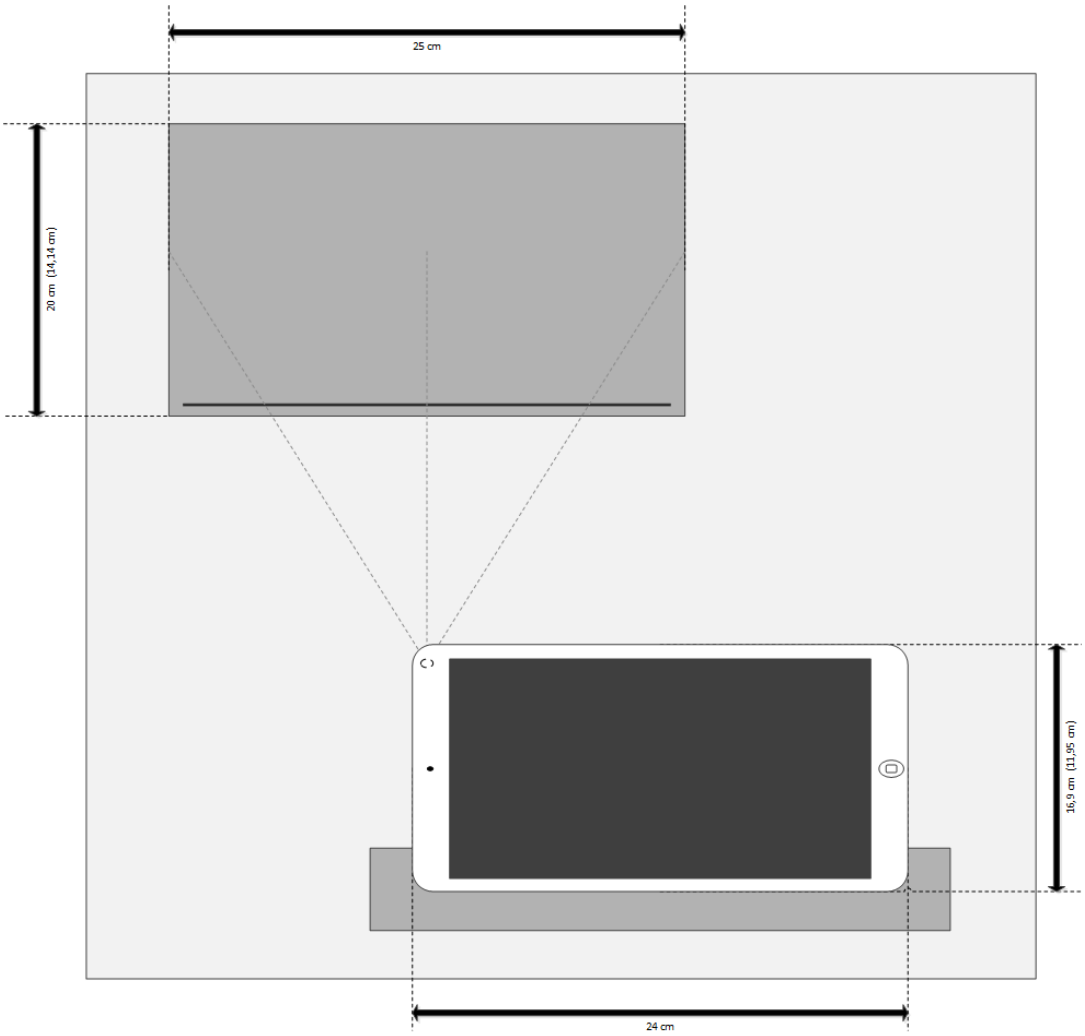


Abbildung 4

Natürlich war die technische Seite ebenfalls bei der Entwicklung zu beachten. Aufgrund der Tatsache, dass die Kamera des Apple iPad an einer oberen Ecke des Gerätes angebracht ist, war es wichtig die Geldablage zur Kamera zu zentrieren, um auf dem Bild auch die gesamte Fläche der Ablage sehen zu können. Außerdem war es erforderlich die Ablage dunkel zu machen, damit die Kanten des Geldscheines für die Kantendetektion gut sichtbar sind. Eine weitere Anforderung an den Aufbau war es, den Bildausschnitt der

Kamera und die Fläche der Ablage so groß zu wählen, dass auch alle Banknoten der zu testenden Währungen in voller Größe Platz finden.

Nach ersten Tests der OCR mit der Vorrichtung, musste an der Beleuchtung für die Geldablage gearbeitet werden. Anfangs war der Aufbau mit keiner Beleuchtung ausgestattet. Dies wurde zuerst mit einem Stück eines LED-Bandes verbessert. Allerdings brachte dies nicht den gewünschten Erfolg, da die Helligkeit nicht ausreichend war. Im späteren wurden einzelne LED-Strahler an die Vorrichtung angebracht, welche Banknoten deutlich besser beleuchteten. Jedoch war die Ausleuchtung nicht gleichmäßig und somit auch nicht geeignet für die OCR. Letztlich wurden mehrere LED-Streifen an dem Aufbau angebracht, die die Geldablage gleichmäßig bestrahlen und somit ein gleichbleibendes Ergebnis der OCR garantieren. Jedoch sollte während eines Votings darauf geachtet werden, dass kein einfallendes Sonnenlicht auf die Ablagefläche fällt, da dadurch die Filtereinstellungen verfälscht werden.

3.2 Funktionsweise (Ablaufdiagramme) der App

Um von Beginn des Projektes eine klare Übersicht der Funktionen und Ausmaße der Applikation zu bekommen, wurden Ablaufdiagramme entwickelt. Diese Ablaufdiagramme beschreiben jeweils einen Automaten und dessen schrittweise auszuführenden Prozesse. Ein Automat ist ein Modell eines Verhaltens, welches aus Zuständen, Zustandsübergängen und Aktionen besteht. Im Programm „Fitness Voter“ wurden zwei Automaten implementiert. Der eine Automat beschreibt alle Nebenfunktionen und der Andere beschreibt einzig die Hauptfunktion der App.

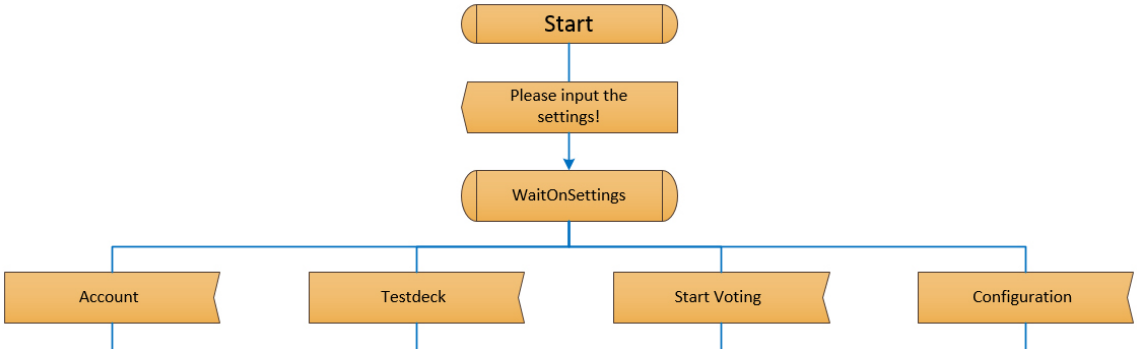


Abbildung 5

Der erste Automat, in welchem die Nebenfunktionen verarbeitet werden, ist der Startautomat. Dieser wird zu Beginn der Ausführung der Applikation aufgerufen und schrittweise durchlaufen. Bei Start der App springt das Programm in den Zustand „Start“ und der User wird mit „Please input the settings!“ aufgefordert Einstellungen vorzunehmen. Daraufhin befindet sich der Startautomat im Zustand „WaitOnSettings“ und wartet auf die Eingabe einer Option. Dem Benutzer werden gleichzeitig 4 mögliche Optionen angeboten

(Account, Testdeck, Start Voting, Configuration). Erst nach Auswahl einer Option wird das Programm fortgeführt. Solange dies nicht der Fall ist, wartet der Automat im Zustand „WaitOnSettings“ auf eine Eingabe (Input).

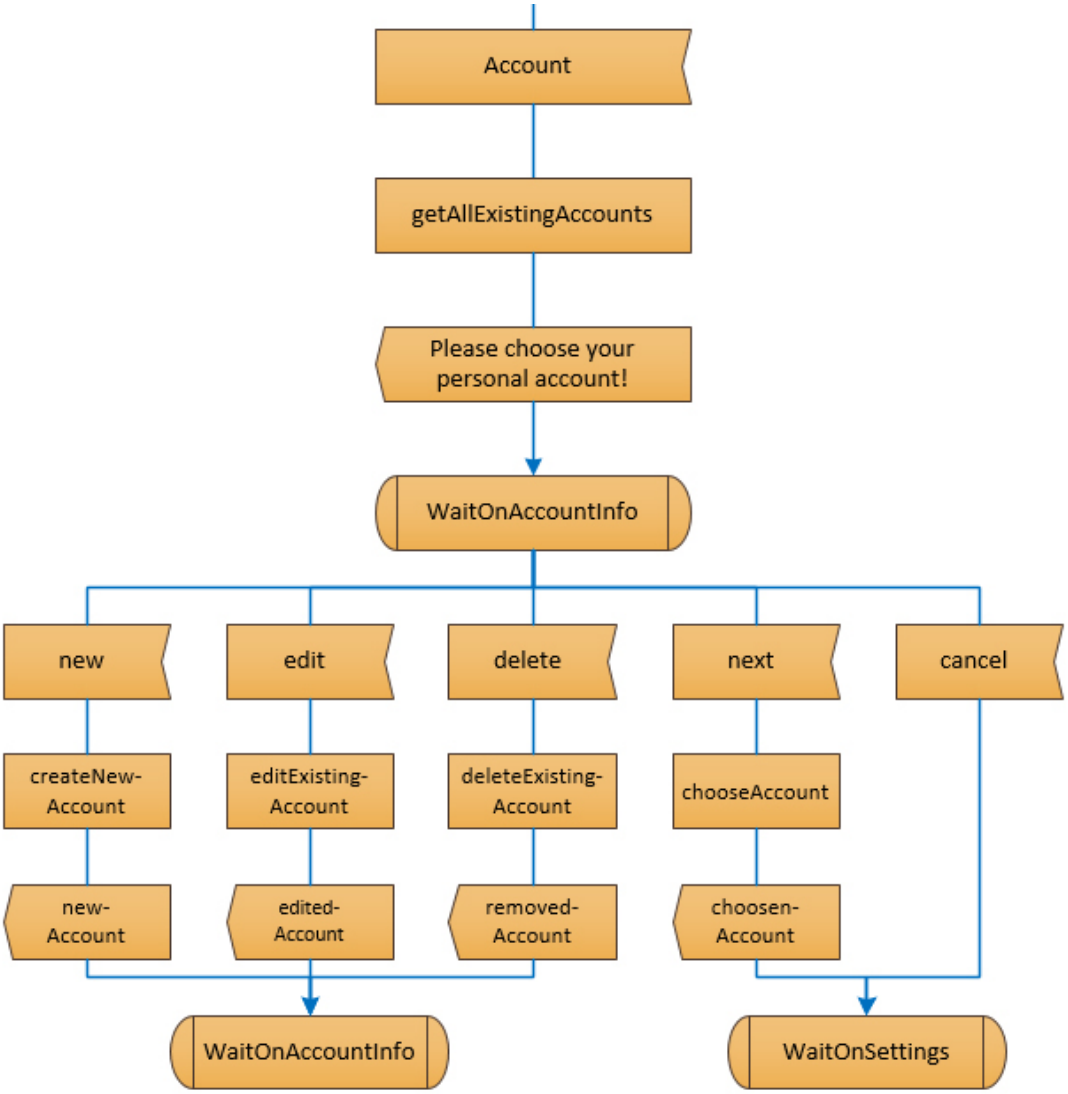


Abbildung 6

Wenn der Startautomat den Input Account bekommt, sollen alle bereits vorhanden Accounts abgerufen und dem User dargestellt werden. Daraufhin wird der Nutzer aufgefordert seinen persönlich Account auszuwählen. Der Automat springt somit in den Zustand „WaitOnAccountInfo“ und wartet wieder auf eine Eingabe durch den Nutzer. Dieser hat die Möglichkeit sich zwischen „new“, „edit“, „delete“, „next“ und „cancel“ zu entscheiden. Bei der Option „new“ werden die eingegebenen Benutzerdaten in der Methode „createNewAccount“ verarbeitet und danach gespeichert. Daraufhin steht der Automat wieder im Zustand „WaitOnAccountInfo“. Mit „edit“ wird ein bereits bestehendes Konto aufgerufen und durch den Nutzer verändert. Diese Änderungen werden verarbeitet und gespeichert und der Automat springt wieder in den letzten Zustand. Wählt der User „delete“ wird der ausgewählte Account gelöscht mittels der Methode „deleteExistingAccount“ und die Daten

in der Datenbank werden aktualisiert. Auch hier steht der Automat nach Abschluss der Bearbeitung wieder im Zustand „WaitOnAccountInfo“. Mittels der Option „next“ wird das ausgewählte Benutzerkonto für den weiteren Programmablauf vorbereitet, für das anstehende Voting in die Datenbank gespeichert und der Automat wieder in den Zustand „WaitOnSettings“ gesetzt. Außerdem hat der User die Möglichkeit „cancel“ zu wählen. Dabei wird lediglich der aktuelle Zustand verlassen und zu „WaitOnSettings“ gewechselt.

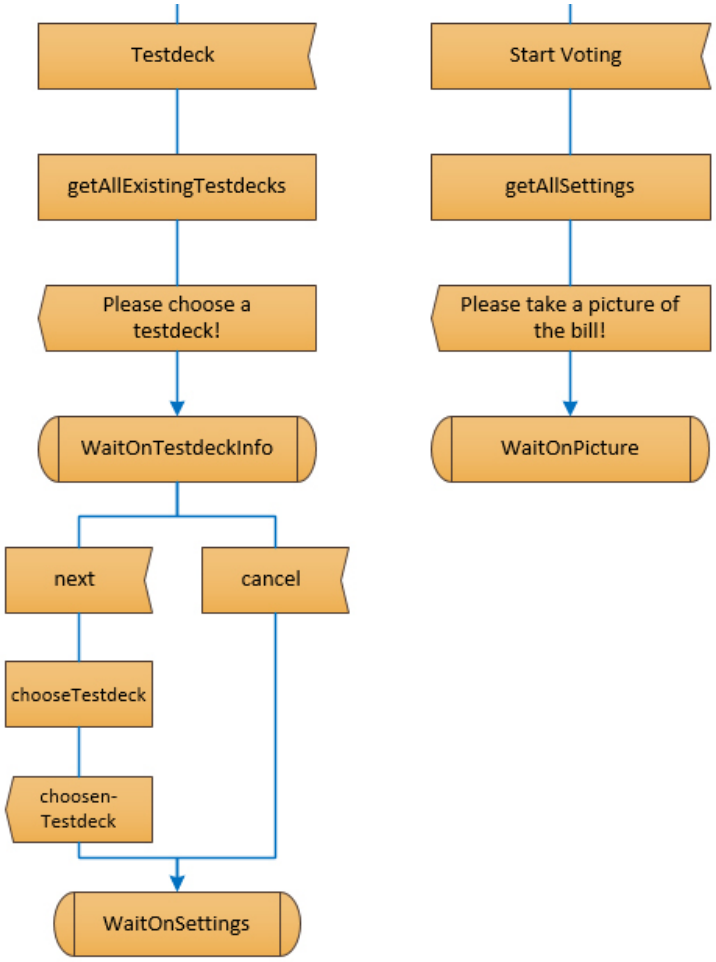


Abbildung 7

Erhält der Startautomat die Eingabe „Testdeck“, werden alle bereits vorhandenen Testdecks aus der Datenbank gelesen und dem User angezeigt. Mit der Meldung „Please choose a testdeck!“ wird der Benutzer zur Auswahl aufgefordert. Der Automat ist nun im neuen Zustand „WaitOnTestdeckInfo“ welcher zwei Input-Möglichkeiten bietet. Wird „next“ aufgerufen, wird das gewählte Testdeck aus der Datenbank entnommen und für das Voting vorbereitet. Daraufhin werden die, für das Bewerten, wichtigen Daten in einer neuen Datenbanktabelle gespeichert und der Zustand wechselt wieder zu „WaitOnSettings“. Mittels „cancel“ wird der Verlauf des Programms abgebrochen und zum Zustand „WaitOnSettings“ gesprungen.

Wird der Input „Start Voting“ betätigt, werden alle bereits vorgenommenen Einstellungen aus den vorbereiteten Datenbanktabellen entnommen und an den zweiten Automaten

(Hauptautomat), welcher für die Hauptfunktion verantwortlich ist, weitergeleitet. Der User wird nun aufgefordert ein Bild zu erzeugen und der Zustand wechselt zu „WaitOnPicture“.

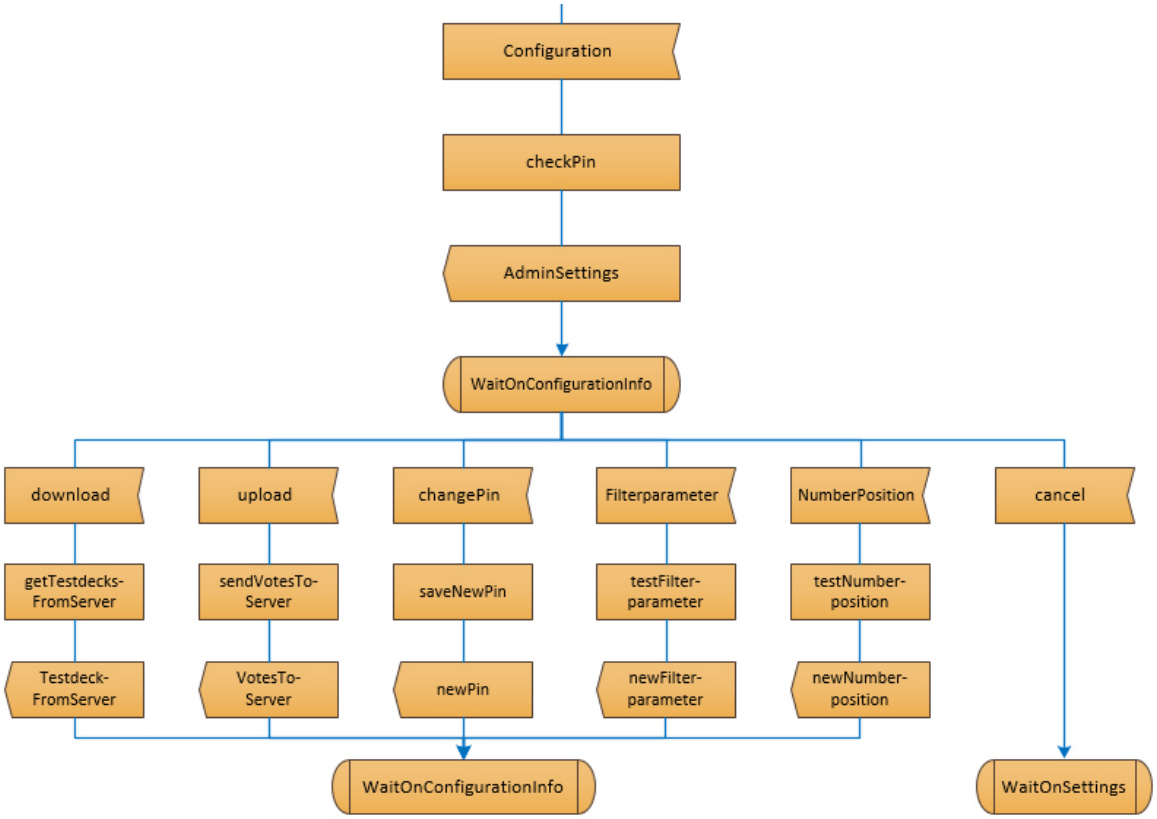


Abbildung 8

Die letzte mögliche Eingabe im Startautomat ist „Configuration“, welche nur mittels PIN fortsetzbar ist, da diese Funktion später nur für den Administrator zugänglich sein soll. Daraus folgt, dass als erstes der PIN auf Korrektheit geprüft werden muss. Wenn dies abgeschlossen und der PIN korrekt ist werden die Administrator-Einstellungen ausgegeben und der Zustand wechselt zu „WaitOnConfigurationInfo“. Erhält der Automat nun den Input „download“, so werden alle vorhandenen Testdecks auf dem Server in einer Liste angegeben und können vom Administrator ausgewählt und heruntergeladen werden. Diese heruntergeladenen Testdecks werden in der Datenbank gespeichert, wobei der Zustand wieder „WaitOnConfigurationInfo“ ist. Bei Aufruf von „upload“ werden sämtliche abgeschlossenen Testdecks aus der Datenbank geladen und in einer Liste angezeigt. Der Anwender kann die hochzuladenden Testdecks einzeln auswählen und zum Webserver hochladen. Der Automat bleibt im Zustand „WaitOnConfigurationInfo“. Die Eingabe „changePin“ öffnet eine neue View, in der der aktuelle PIN einmal und der neue PIN zweimal einzugeben ist und daraufhin geändert wird. Der PIN zum Einwählen in das Configuration-Menü wird ebenfalls in der CoreData-Datenbank gespeichert. Der Zustand bleibt unverändert. Mittels des Input „Filterparameter“ wird eine View geöffnet, in der eine Live Ansicht der Kamera zu sehen ist. Auf diese Ansicht können verschiedene Filter angewendet werden um die Parameter für ein neues Testdeck bestimmen zu können. Die jeweiligen Werte der eingestellten Filterparameter werden mittels Labels ausgegeben. Nach Abschluss

der Tests bleibt der Zustand des Automaten „WaitOnConfigurationInfo“. Der Input „Numberposition“ ist so ähnlich wie der Input „Filterparameter“. Hier werden lediglich statt der Filterparameter-Werte, Schwellwerte für die Kantendetektion der Banknote und Koordinaten für den Bereich der Seriennummern getestet. Diese sind ebenfalls ein wichtiger Bestandteil eines Testdecks und werden per Label ausgegeben. Auch hier bleibt der Zustand „WaitOnConfigurationInfo“ erhalten. Lediglich bei „cancel“ wird der Zustand zu „WaitOnSettings“ gewechselt.

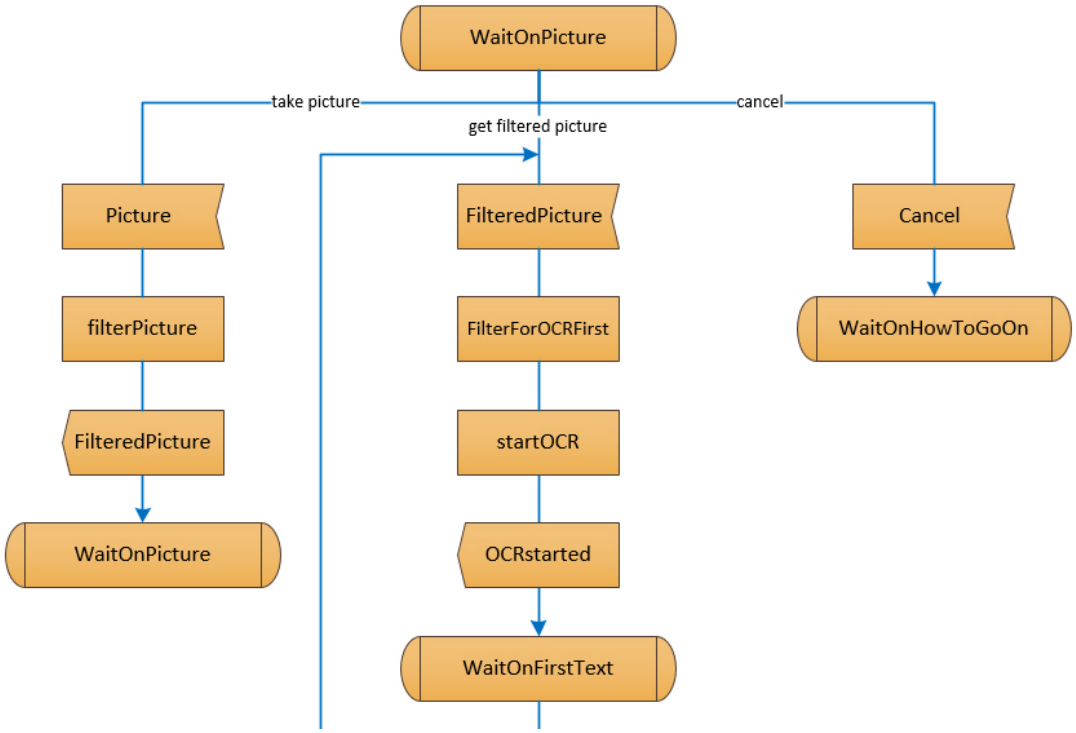


Abbildung 9

Der Hauptautomat beginnt im Zustand „WaitOnPicture“ nachdem im Startautomat der Input „Start Voting“ gewählt wurde. Wird nun der Input „Picture“ an den Automaten übergeben, wird ein Bild erzeugt und in der Methode „filterPicture“ mittels der Schwellwerte für die Kantendetektion aus dem Testdeck gefiltert. Dabei wird das gefilterte Bild zurückgegeben und der Zustand „WaitOnPicture“ bleibt bestehen. Somit wird automatisch der nächste Input „FilteredPicture“ aufgerufen. In diesem Zweig des Ablaufdiagramms werden die Filterparameter aus dem Testdeck dafür verwendet, um das Bild für die Texterkennung der ersten Seriennummer vorzubereiten und zu filtern. Außerdem wird die OCR mit den Koordinaten für den Bereich der Texterkennung gestartet und beginnt damit, den Text vom Bild auszulesen. Dies wird sichtbar anhand eines Fortschrittsbalkens und der Zustand ändert sich zu „WaitOnFirstText“. Die dritte mögliche Eingabe ist „Cancel“. Hierbei wird lediglich der Zustand verändert und auf „WaitOnHowToGoOn“ gesetzt.

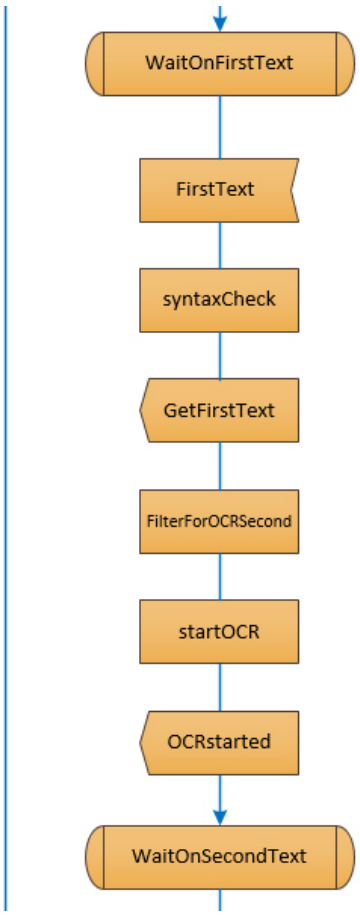


Abbildung 10

Im Zustand „WaitOnFirstText“ ist nur ein Input „FirstText“ möglich. Dabei wird das Ergebnis der Texterkennung an die Methode „syntaxCheck“ geliefert und ausgewertet. Hierbei wird überprüft, ob die Seriennummer der Länge und Syntax der Vorgabe aus dem Testdeck entspricht. Zusätzlich liefert die OCR einen Prozentwert zu jedem ausgelesenem Zeichen, welcher in einem Array gespeichert wird. Diese Prozentwerte werden benutzt um eine Seriennummer zusammenzusetzen, welche die aktuell höchsten Prozentwerte besitz. Nach Bestehen dieses Syntaxvergleiches wird die Nummer in einem Array gespeichert.

Außerdem wird das Bild mittels der Filter für die zweite Seriennummer vorbereitet und gefiltert. Des Weiteren wird auch hier die Position und Größe des auszulesenden Bereiches aus dem Testdeck gelesen und übergeben. Danach wird die OCR gestartet und der Fortschrittsbalken aktualisiert. Der Zustand des Hauptautomaten wechselt zu „WaitOnSecondText“.

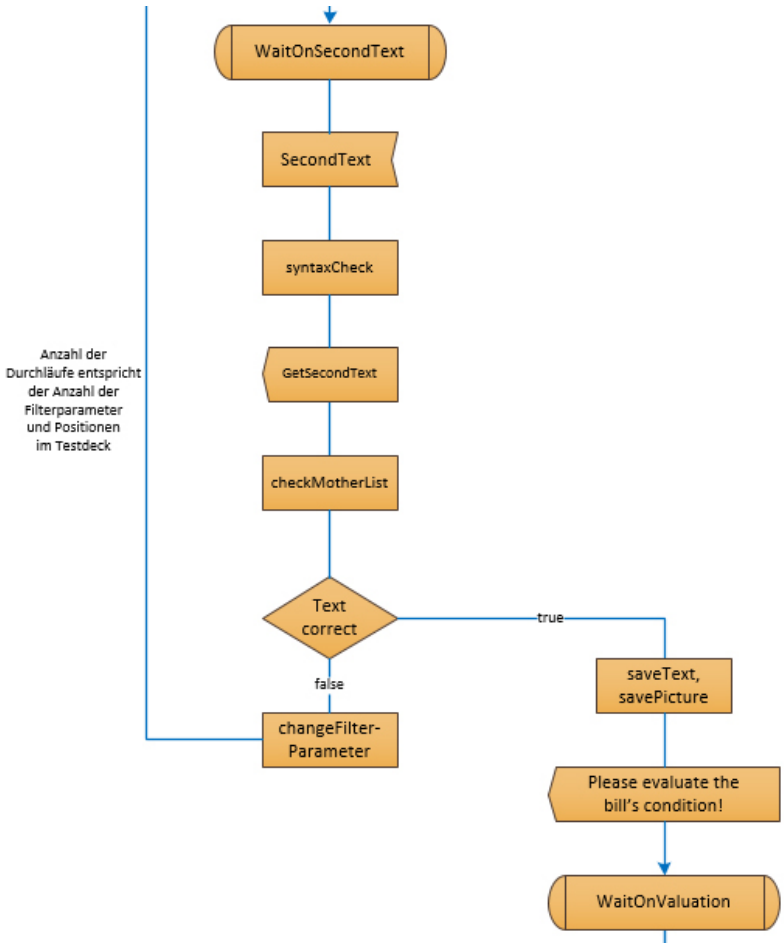


Abbildung 11

Im Zustand „WaitOnSecondText“ kann ebenfalls wieder nur ein Input „SecondText“ erfolgen. Wenn dies geschehen ist, wird die zweite ausgelesene Seriennummer zurückgeliefert. Nun muss auch diese Nummer den Syntaxvergleich durchlaufen. Wird dieser bestanden kann auch dieses OCR-Ergebnis zur Weiterverarbeitung in ein Array gespeichert werden. Anschließend wird die Methode „checkMotherList“ aufgerufen, in der die ausgelesenen Seriennummern geprüft werden.

Wenn nach der Prüfung dieser OCR-Ergebnisse keine korrekte Seriennummer gefunden wurde, werden die Filterparameter verändert und der Zustand wechselt zu „WaitOnPicture“, wobei der Input „FilteredPicture“ mitgeliefert wird und somit das Auslesen der Nummer neu gestartet wird. Dieser Vorgang ist so oft durchführbar, wie Filterparameter und mögliche Positionen der Seriennummer vorhanden sind.

Wird jedoch eine korrekte Seriennummer zurückgeliefert, wird der ermittelte Text mit dem dazugehörigen Bild gespeichert. Der User wird aufgefordert die Banknote zu bewerten und der Zustand des Hauptautomaten wechselt zu „WaitOnValuation“.

Um zu verstehen, wann eine korrekte Seriennummer zurückgeliefert wird, muss man sich die Methode „checkMotherList“ etwas genauer anschauen. Dies wird in den folgenden Abbildungen deutlich.

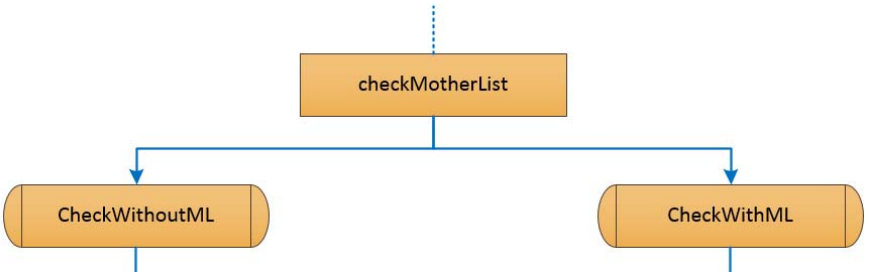


Abbildung 12

In der Methode „checkMotherList“ wird zuerst überprüft, ob im aktuellen Testdeck Informationen zu einer Mutterliste vorhanden sind. Je nach Ergebnis erhält der Automat einen neuen Zustand. Wenn keine Mutterliste vorhanden ist, wechselt der Zustand zu „CheckWithoutML“ ansonsten zu „CheckWithML“.

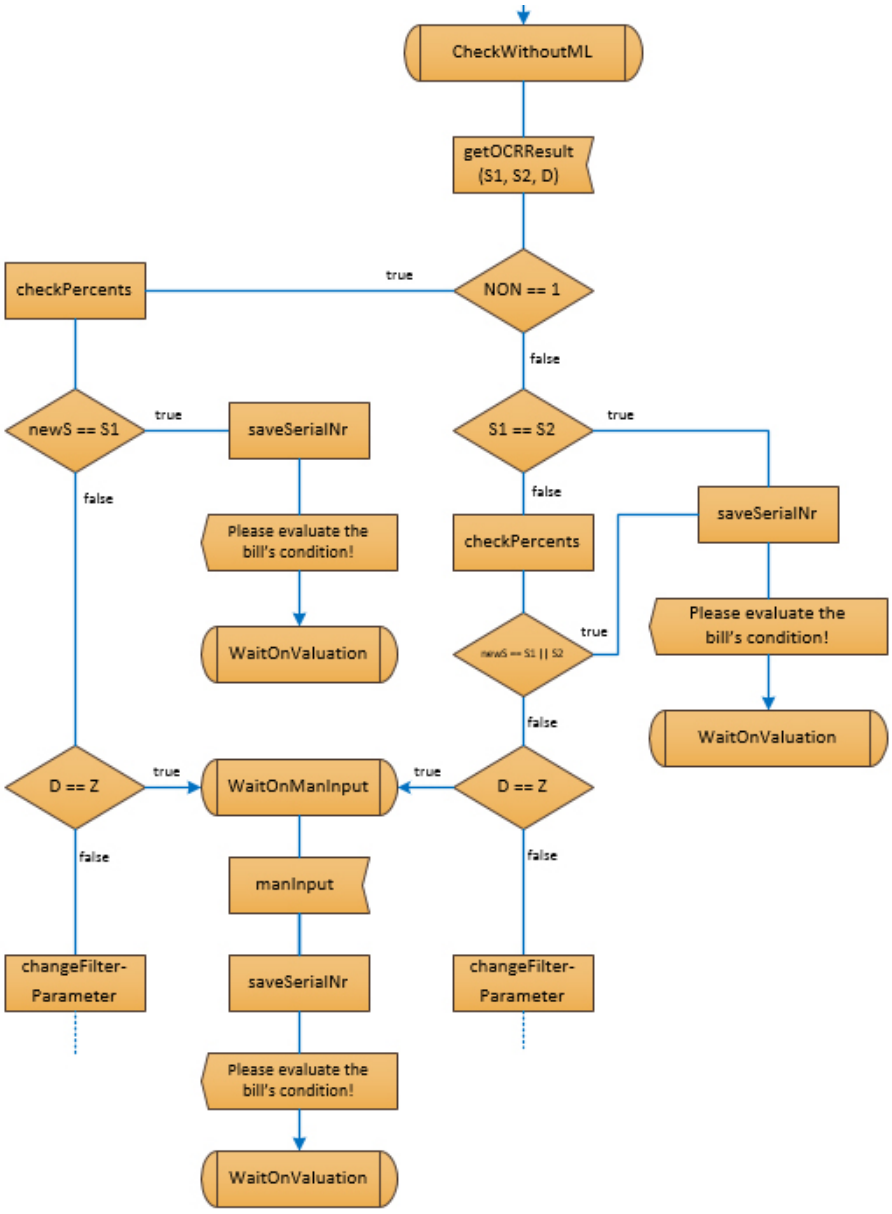


Abbildung 13

Wenn keine Mutterliste zum ausgewählten Testdeck vorhanden ist, steht der Hauptautomat im Zustand „CheckWithoutML“. An dieser Stelle ist nur ein Input möglich, wobei die OCR-Ergebnisse übergeben werden. Der Automat prüft die Anzahl der Seriennummer „NON“, welche auf der Banknote vorhanden sind. Wenn auf der Banknote nur eine Nummer auszulesen ist, wird in der Methode „checkPercents“ ein neuer String gebildet, der sich aus den Zeichen mit den höchsten Prozentwerten der jeweiligen Stelle zusammensetzt. Dabei müssen die Prozentwerte oberhalb der 80 Prozent Marke liegen.

Dieser neugebildete String wird mit den OCR-Ergebnissen verglichen. Bei Übereinstimmung wird die ausgelesene Nummer gespeichert und der Automat springt in den Zustand „WaitOnValuation“, wobei der User aufgefordert wird die Banknote zu bewerten. Sollte dies nicht der Fall sein, da die Strings nicht übereinstimmen, wird die Anzahl der Durchläufe geprüft. Die maximale Anzahl der Durchläufe „Z“ ergibt sich aus der Anzahl der Filterparameter mal der Anzahl der Positionen des auszulesenden Rechtecks aus dem Testdeck. Entspricht der aktuelle Durchlauf der Maximalanzahl aller Durchläufe, wechselt der Zustand des Hauptautomaten auf „WaitOnManInput“. Ansonsten werden die Filterparameter verändert und der Zustand wechselt zu „WaitOnPicture“. Im Zustand „WaitOnManInput“ bekommt der Automat die vom User eingegebene Seriennummer und speichert diese. Der Automat wechselt dann zu „WaitOnValuation“.

Entspricht die Anzahl der Seriennummern „NON“ nicht eins, so hat der Automat mehrere OCR-Ergebnisse zum Vergleichen. Stimmen diese Nummern überein, wird die Nummer gespeichert und der Zustand wechselt zu „WaitOnValuation“. Ansonsten wird die Methode „checkPercents“ aufgerufen, welche wieder einen neuen String zusammensetzt, der auch der Syntax entsprechen muss. Danach wird dieser String mit beiden OCR-Ergebnissen verglichen. Wird dabei eine Übereinstimmung gefunden wird diese Nummer gespeichert und der Zustand wechselt auch hier zu „WaitOnValuation“. Wenn nicht, dann wird die Anzahl der Durchläufe geprüft. Bei Erreichen des Enddurchlaufes ändert sich der Zustand des Automaten zu „WaitOnManInput“, andernfalls werden die Filterparameter verändert und der Zustand springt wieder zu „WaitOnPicture“.

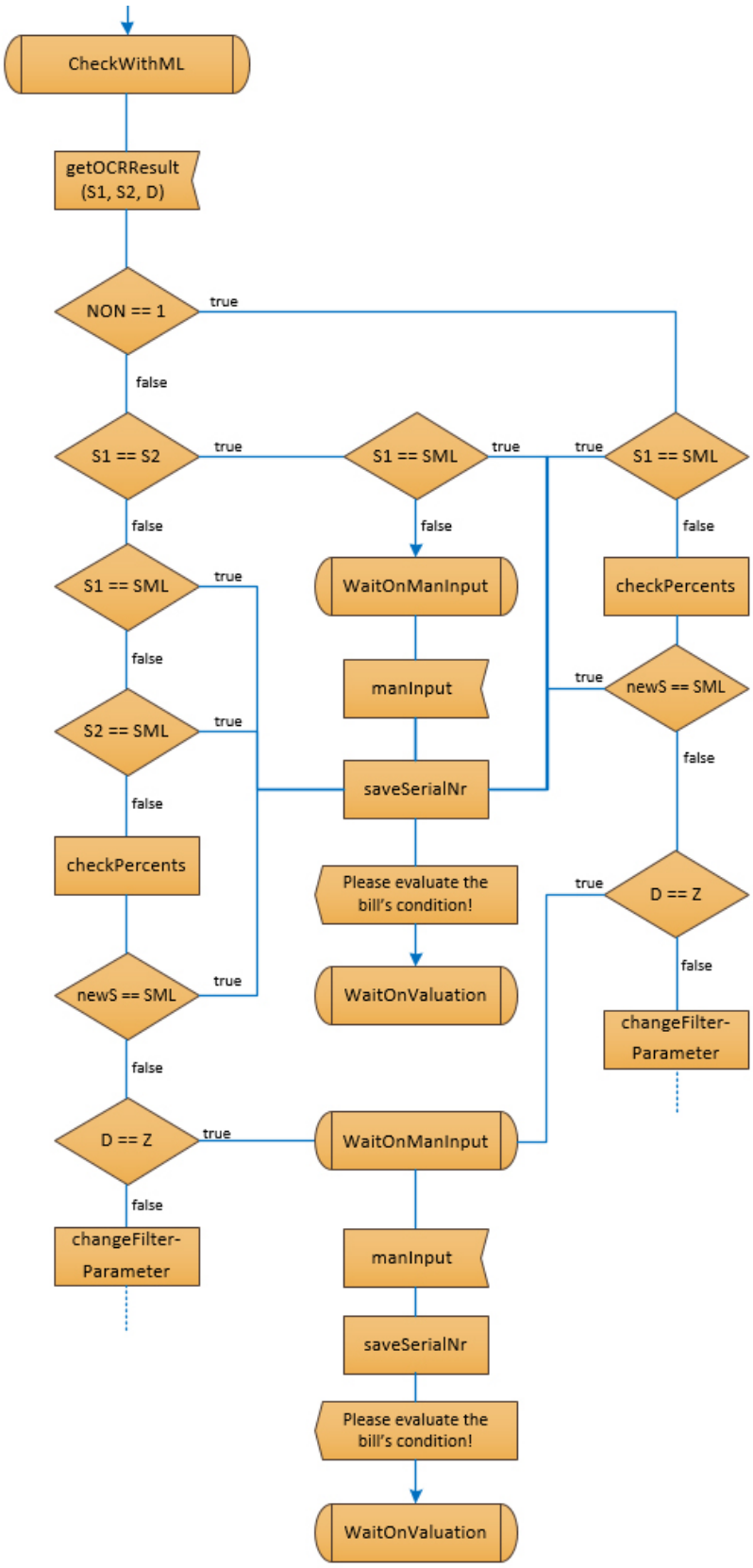


Abbildung 14

Wenn jedoch eine Mutterliste zu dem Testdeck vorhanden ist, befindet sich der Automat im Zustand „CheckWithML“. Als Input bekommt er die OCR-Resultate und kann nun diese

auf Korrektheit prüfen. Dabei liegt die Vorgehensweise wieder darin, die Anzahl der Seriennummern auf der Note zu prüfen.

Wenn nur eine Nummer vorhanden ist wird der, von der Texterkennung gelieferte String mit allen Werten der Mutterliste verglichen. Sollte das Ergebnis der OCR in der Mutterliste vorhanden sein, wird diese Nummer gespeichert und der Zustand des Automaten wechselt zu „WaitOnValuation“. Ist das jedoch nicht der Fall, wird die „checkPercents“ Methode aufgerufen, in der wieder ein String zusammengestellt wird. Dieser String wird anschließend ebenfalls mit der Mutterliste abgeglichen. Sollte es dabei zu einer Übereinstimmung kommen, wird der erstellte String als Seriennummer gespeichert und der Zustand ist dann „WaitOnValuation“. Ansonsten werden die Filterparameter verändert und der nächste Durchlauf wird gestartet indem der Zustand zu „WaitOnPicture“ wechselt.

Für den Fall, dass zwei Seriennummern auf dem Geldschein sind, wird die erste mit der zweiten Nummer verglichen. Bei Gleichheit dieser ausgelesenen Seriennummern wird noch überprüft, ob diese auch in der Mutterliste enthalten ist. Wenn ja, so wird die Seriennummer gespeichert und der Zustand ändert sich zu „WaitOnValuation“, wenn nicht, dann springt der Zustand auf „WaitOnManInput“.

Sollten die beiden OCR-Resultate nicht übereinstimmen, wird die erste erhaltene Nummer mit der Mutterliste verglichen. Ist diese darin enthalten wird gespeichert, ansonsten wird die zweite Nummer der OCR mit der Mutterliste verglichen. Wenn diese in der Mutterliste enthalten ist, wird sie als Seriennummer gespeichert. Bei beiden Speichervorgängen würde sich der Zustand zu „WaitOnValuation“ ändern. Führen jedoch beide Versuche nicht zu Übereinstimmungen, wird mit der Methode „checkPercents“ ein neuer String erzeugt und mit der Mutterliste verglichen. Sollte dieser in der Liste enthalten sein wird er als Seriennummer gespeichert und der Zustand „WaitOnValuation“ gesetzt.

Ist auch nach diesem Vergleich kein Treffer gefunden, wird die Anzahl der Durchläufe überprüft. Wenn der letzte Durchlauf erreicht ist, wechselt der Zustand zu „WaitOnManInput“. Sind jedoch noch mehr Durchläufe möglich, werden die Filterparameter geändert und der Zustand des Hauptautomaten wird auf „WaitOnPicture“ gesetzt.

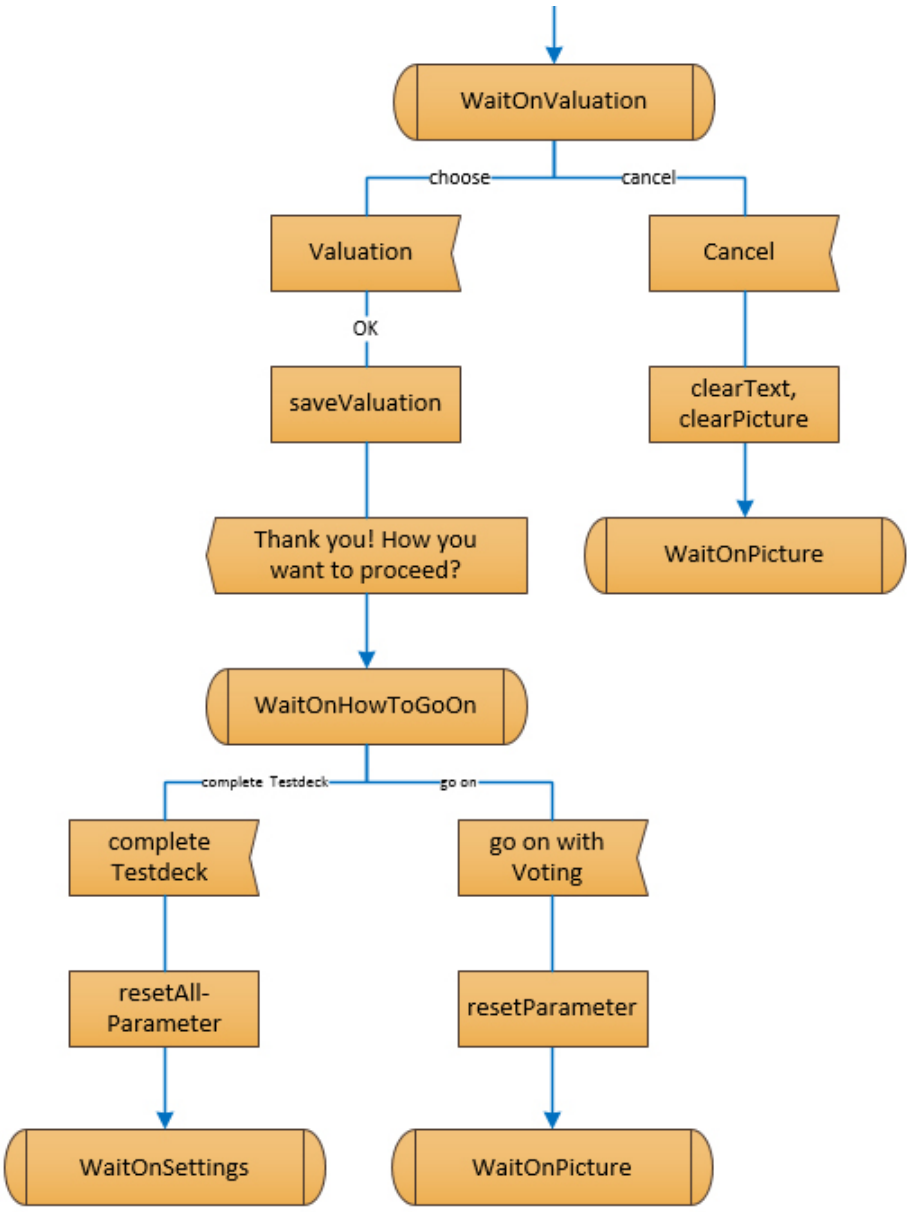


Abbildung 15

Befindet sich der Automat im Zustand „WaitOnValuation“ sind zwei Inputs möglich. Wird eine Bewertung zum Geldschein abgegeben, so wird die Methode „saveValuation“ aufgerufen. In dieser Methode wird die Bewertung zum jeweiligen Bild und der jeweiligen Seriennummer gespeichert. Anschließend wird eine neue View ausgegeben und der Nutzer wird gefragt wie er fortfahren möchte. Dabei ändert sich der Zustand des Hauptautomaten zu „WaitOnHowToGoOn“.

Bricht jedoch der User den Vorgang ab und übergibt somit den Input „Cancel“, werden die Methoden „clearText“ und „clearPicture“ aufgerufen. In diesen Methoden werden die bereits gesicherten Daten der aktuellen Banknote gelöscht. Daraufhin wird der Zustand auf „WaitOnPicture“ gesetzt.

Im Zustand „WaitOnHowToGoOn“ sind ebenfalls wieder zwei Inputs möglich. Zum einen kann der Benutzer mittels „complete Testdeck“ das Testdeck abschließen. Dabei werden alle Parameter, welche für das Voting notwendig sind, zurückgesetzt und der Zustand wechselt zu „WaitOnSettings“. Zum anderen hat der Nutzer die Möglichkeit mit dem Voting fortzufahren. In diesem Fall erhält der Automat den Input „go on with Voting“ und führt dabei die Methode „resetParameter“ aus. In dieser Methode, werden lediglich die Parameter zurückgesetzt, welche während des Votings verändert wurden. Dies sind zum Beispiel die Anzahl der Durchläufe, die Filterparameter oder die Prozente des gebildeten Strings. Anschließend verändert sich der Zustand zu „WaitOnPicture“.

4 Implementierung der App mit Objective C

Aufbauend auf dem entwickelten Ablauf, wurde die Applikation umgesetzt. Doch bevor etwas implementiert werden konnte, mussten die Frameworks der Erweiterungen (GPUImage, Tesseract und CoreData) integriert werden, wobei unter anderem auch mit Pods gearbeitet wurde.

Natürlich war es wichtig die Strukturen eines Automaten einzuhalten. Alle, in den Ablaufdiagrammen vorhandenen Zustände mussten angelegt und mit Funktionen gefüllt werden. Durch die Integration der Erweiterungen werden viele Methoden mitgeliefert, was das füllen der Zustände etwas einfacher gestaltet.

Um einen optimalen Ablauf der Applikation zu gewährleisten, mussten die, von den Erweiterungen mitgelieferten Methoden natürlich etwas angepasst werden. Außerdem war es notwendig neue Methoden anzulegen, um gewisse Probleme zu lösen. Im Folgenden werden die Wichtigsten davon kurz beschrieben.

4.1 Kantendetektion

Die Kantendetektion wird direkt nach dem Erzeugen des Bildes mit der Banknote gestartet. Sie wird im Hauptautomat im Zustand „WaitOnPicture“ aufgerufen und bekommt dabei das Bild übergeben. Anschließend wird als erstes das Bild herabskaliert und mittels eines SobelEdgeDetectionFilters, welcher Teil des GPUImage-Frameworks ist, für die Ermittlung der Kanten vorbereitet. Diese gefilterte Grafik wird dann an die Methode „getImageWithEdgeLineFromImage“ gesendet.

In dieser Methode sollen die Kanten der Banknote ermittelt werden. An diese Kanten sollen rote Geraden gezeichnet werden, um später in pixelweisen Schritten nach dem ersten roten Pixel suchen zu können.

```
- (void)getImageWidthEdgeLineFromImage:(GPUImagePicture*)pictureInput
                       withLineDetector:(GPUImageHoughTransformLineDetector *)lineDetector
                       withLineColorRed:(CGFloat)red withLineColorGreen:(CGFloat)green withLineColorBlue:(CGFloat)blue
                          withLineWidth:(CGFloat)lineWidth withEdgeThreshold:(CGFloat)edgeThreshold
              withLineDetectionThreshold:(CGFloat)lineDetectionThreshold
{
    [pictureInput removeAllTargets];
    [pictureInput addTarget:lineDetector];

    [lineDetector setEdgeThreshold:edgeThreshold];
    [lineDetector setLineDetectionThreshold:lineDetectionThreshold];
    [lineDetector setLinesDetectedBlock:^(GLfloat* lineArray, NSUInteger linesDetected, CMTime frameTime){
        NSLog(@"Number of lines: %ld", (unsigned long)linesDetected);

        GPUImageLineGenerator *lineGenerator = [[GPUImageLineGenerator alloc] init];
        [lineGenerator setLineColorRed:red green:green blue:blue];
        [lineGenerator setLineWidth:lineWidth];
        [lineGenerator forceProcessingAtSize:[pictureInput outputImageSize]];

        GPUImageAlphaBlendFilter *blendFilter = [[GPUImageAlphaBlendFilter alloc] init];
        [blendFilter setMix:1.0];
        [blendFilter forceProcessingAtSize:[pictureInput outputImageSize]];

        [pictureInput addTarget:blendFilter];

        [lineGenerator addTarget:blendFilter];

        [blendFilter useNextFrameForImageCapture];

        [lineGenerator renderLinesFromArray:lineArray count:linesDetected frameTime:frameTime];

        dispatch_async(dispatch_get_main_queue(), ^{
            filteredImageWidthEdgeLine = [blendFilter imageFromCurrentFramebuffer];
            [lineGenerator removeAllTargets];
            [delegate edgeMarkingComplete];
        });

    }];
    [pictureInput processImage];

}
```

Abbildung 16

Als erstes bekommt die Methode beim Aufruf einige Parameter übergeben. Neben einem „GPUImageHoughTransformLineDetector“, welcher die Linien im Bild mittels der Hough Transformation ermittelt, werden auch Farbwerte und Linienbreite für die einzuzeichnenden Linien übergeben. Außerdem werden Schwellwerte übergeben, welche dem LineDetector zugewiesen werden.

Der LineDetector wird dem pictureInput zugewiesen, welcher das gefilterte Bild beinhaltet. Anschließend werden dem LineDetector die beiden Schwellwerte übergeben und mittels „setLinesDetectedBlock“ die Kantenermittlung gestartet.

Zum Zeichnen der Linien wird ein „GPUImageLineGenerator“ benötigt. Dieser bekommt den Farbwert der Linienfarbe, die Breite der Linie und die Bildgröße für die Ausgabe übergeben.

Um diese Linien auch sichtbar zu machen, wird mit Hilfe eines „GPUImageAlphaBlendFilters“ das Ausgangsbild mit dem Bild, auf dem die Linien eingezeichnet wurden, ersetzt. Dieser Filter wird auf pictureInput und dem lineGenerator angewendet. Da das Bild aber erst später mit dem Filter gecaptured werden soll, wartet der Filter dank „useNextFrameForImageCapture“ auf seinen Einsatz.

Anschließend erhält der lineGenerator die Linienkoordinaten und Anzahl vom lineDetector. Danach wird an das UIImage „filteredImageWithEdgeLine“ das Bild übergeben und der blendFilter aufgerufen, womit nun ein Bild mit roten Geraden an den Kanten entstanden ist. Abschließend wird die delegate-Methode „edgeMarkingComplete“ aufgerufen.

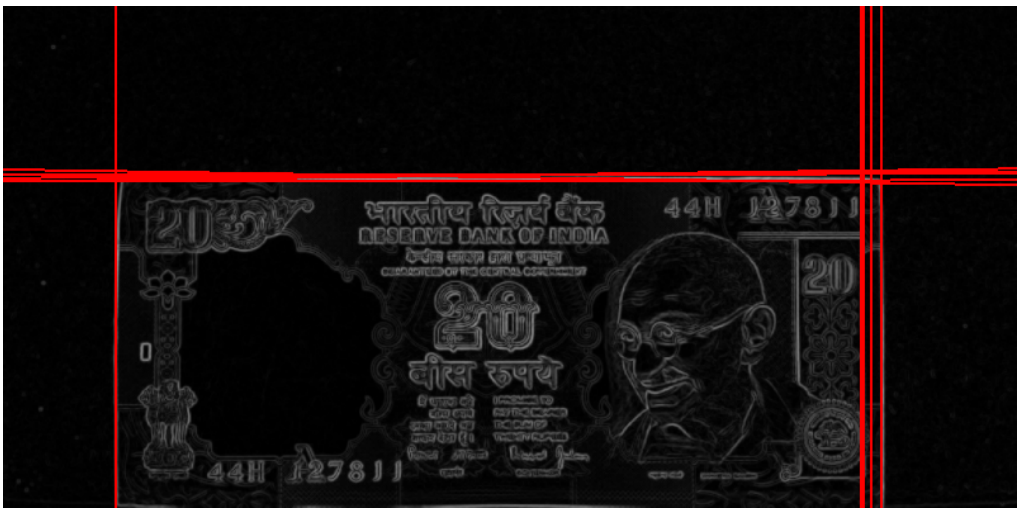


Abbildung 17

In der delegate-Methode „edgeMarkingComplete“ werden dann die Koordinaten und die Breite und Höhe der Banknote ermittelt. Dies wird mittels pixelweisem Auswerten der Pixelfarbe realisiert. Sobald das erste rote Pixel gefunden wird, wird das Abschreiten beendet und der Wert gespeichert. Dieses Verfahren wird von allen vier Seiten des Geldscheines ausgeführt.

4.2 Ermittlung der Y-Koordinate und der exakten Höhe einer Seriennummer

Um die Position der Seriennummer möglichst genau zu bestimmen und somit das beste OCR-Ergebnis zu erhalten, soll die Region, welche im Testdeck für die jeweilige Nummer angegeben ist, durchsucht werden. Dieser Vorgang wird direkt nach der Kantendetektion ausgeführt.

Da das Ermitteln der Y-Koordinate und der Höhe zeilenbasiert und das Ermitteln der X-Koordinate und der Breite spaltenbasiert ist, müssen zwei unterschiedliche Methoden verwendet werden.

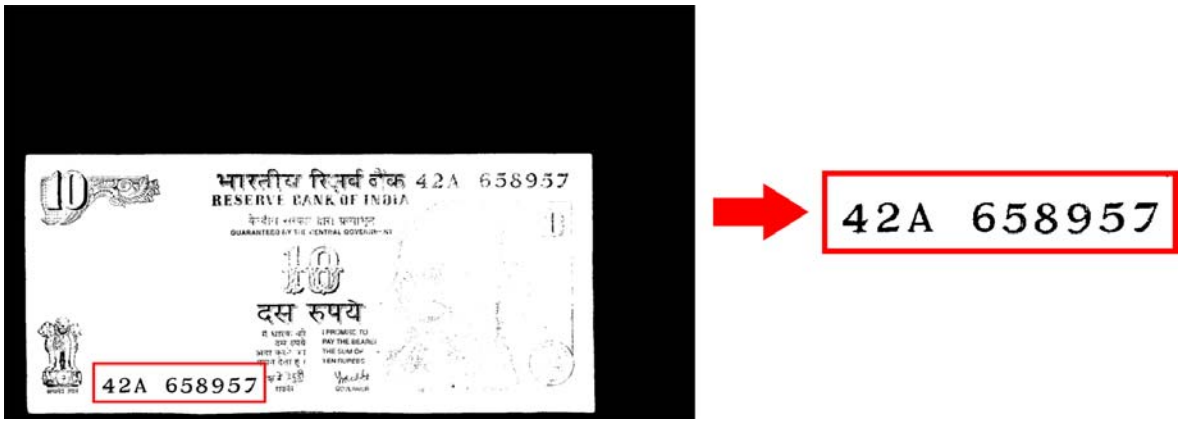


Abbildung 18

Als Vorbereitung für die Methode „getBlackPixelFromLeftWithRed“, welche für die Höhe und Y-Koordinate der Seriennummer zuständig ist, wird ein Bildausschnitt vom gefilterten Gesamtbild erzeugt. Dieser Bildausschnitt entspricht der Region der jeweiligen Nummer, die im Testdeck angegeben ist.

```
- (id) getBlackPixelFromLeftWithRed:(UInt8)red withBlue:(UInt8)blue withGreen:(UInt8)green
                          withAlpha:(UInt8)alpha withImage:(UIImage*)image
{
    CFDataRef pixelData = CGDataProviderCopyData(CGImageGetDataProvider(image.CGImage));
    const UInt8* data = (UInt8 *) CFDataGetBytePtr(pixelData);
    int imageWidth = (int)CGImageGetWidth(image.CGImage);
    int imageHeight = (int)CGImageGetHeight(image.CGImage);
    int pixelX = imageWidth-1; // The X coordinate of the pixel you want to retrieve
    int pixelY = imageHeight-1; // The Y coordinate of the pixel you want to retrieve
    int blackPixelPerRow = 0;
    int NumberStartY = -1;
    int blackLineCounter = 0;
    NSMutableArray *bPixelPerRowArray = [[NSMutableArray alloc] init];;
    UInt8 _red,_green,_blue,_alpha;

    for (int i=0; i<pixelY; i++) {
        int counter=0;
        while (counter < pixelX) {
            int pixelInfo = ((imageWidth  * i) + counter ) * 4;
            if ([[[UIDevice currentDevice] model] isEqualToString:@"iPad Simulator"]) {
                _red   = data[pixelInfo + 0];
                _green = data[pixelInfo + 1];
                _blue  = data[pixelInfo + 2];
                _alpha = data[pixelInfo + 3];
            }else{
                _blue   = data[pixelInfo + 0];
                _green = data[pixelInfo + 1];
                _red  = data[pixelInfo + 2];
                _alpha = data[pixelInfo + 3];
            }
            if (_red == red && _green == green && _blue == blue && _alpha == alpha) {
                blackPixelPerRow++;
            }
            counter++;
        }
        if (blackPixelPerRow >= 25) {
            if (NumberStartY == -1){
                NumberStartY = i-8;
            }
            blackLineCounter++;
        }else{
            if (blackLineCounter>30) {
                [bPixelPerRowArray addObject:[NSNumber numberWithInt:NumberStartY]];
                [bPixelPerRowArray addObject:[NSNumber numberWithInt:(blackLineCounter+16)]];
                break;
            }
            blackLineCounter = 0;
            NumberStartY = -1;
        }
        blackPixelPerRow = 0;
    }
    CFRelease(pixelData);
    return bPixelPerRowArray;
}
```

Abbildung 19

Die Methode „getBlackPixelFromLeftWithRed“ wird im Zustand „WaitOnPicture“ des Hauptautomaten aufgerufen. Dabei werden Parameter für die RGB-Werte und den Alpha-Wert übergeben. Außerdem wird der Bildausschnitt, in dem sich die Seriennummer befindet, mitgeliefert.

Als erstes wird aus dem Bildausschnitt mittels „CGDataProviderCopyData“ eine „CFDataReferenz“ erzeugt. Daraus wird mit Hilfe von „CFDataGetBytePtr“ ein UInt8 gebildet, welcher für jedes Pixel des Bildes einen Rot-, Grün-, Blau- und Alphawert beinhaltet.

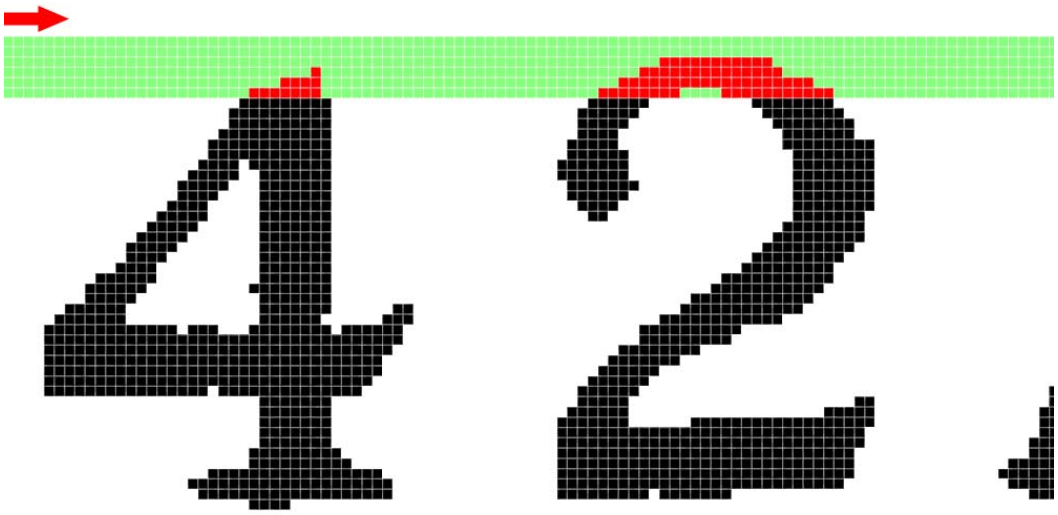


Abbildung 20

Innerhalb einer „for“-Schleife werden alle Pixel zeilenweise, oben beginnend, von der linken Bildkante aus geprüft. Entsprechen die drei Farbwerte gleich null und der Alphawert ist gleich 1, so ist das Pixel schwarz. Für jedes schwarze Pixel in der Zeile wird der Counter „blackPixelPerRow“ um 1 hochgezählt. Wurde eine Zeile komplett durchlaufen, wird dieser Counter geprüft. Sind in der Zeile 25 oder mehr schwarze Pixel enthalten, so ist es sehr wahrscheinlich, dass diese Zeile zur Seriennummer gehört.

Nun wird noch geprüft, ob es die erste Zeile ist in der so viele schwarze Punkte gefunden wurden. In diesem Fall wird die Zeilennummer als Y-Koordinate verwendet. Um eventuell wegfallende Pixel (durch Schräglage der Banknote) zu erhalten, wird ein Puffer von 8 Pixel gegeben. Also entspricht die Y-Koordinate der Zeilennummer minus 8 Pixel als Puffer.

Des Weiteren wird ein Counter „blackLineCounter“ für die Anzahl der aufeinander folgenden Zeilen mit 25 oder mehr schwarzen Punkten hochgezählt. Dieser Counter wird immer überprüft, wenn eine Zeile mit weniger als 25 schwarzen Pixeln ausgelesen wurde. Ist also die Anzahl der eventuell zur Seriennummer gehörenden Zeilen weniger oder gleich 30, so wird dieser Counter auf null gesetzt und der Wert für die Y-Koordinate wird ebenfalls zurückgesetzt. Sollte der „blackLineCounter“ allerdings größer 30 sein, so entspricht der Wert des Counters gleich der Höhe. Aber um bei der Höhe die eventuell auch wegfallenden Pixel zu erhalten, wird ebenfalls ein Puffer gegeben. Dieser beinhaltet die 8 Pixel von der Y-Koordinate und wird um weitere 8 Pixel, für die untere Kante der Seriennummer, erweitert. Somit ergibt sich die Höhe aus dem Wert des „blackLineCounter“ plus 16 Pixel.

Die Höhe und die Y-Koordinate werden anschließend in ein Array gespeichert und zum Abschluss der Methode an den Automaten zurückgegeben.

4.3 Ermittlung der X-Koordinate und der exakten Breite einer Seriennummer

Nachdem die Höhe und die Y-Koordinate ermittelt wurden, fehlt noch die Breite und X-Koordinate der Seriennummer. Dabei wird eine ähnliche Vorgehensweise angewendet, mit der Ausnahme, dass diese Methode spaltenbasiert ist.

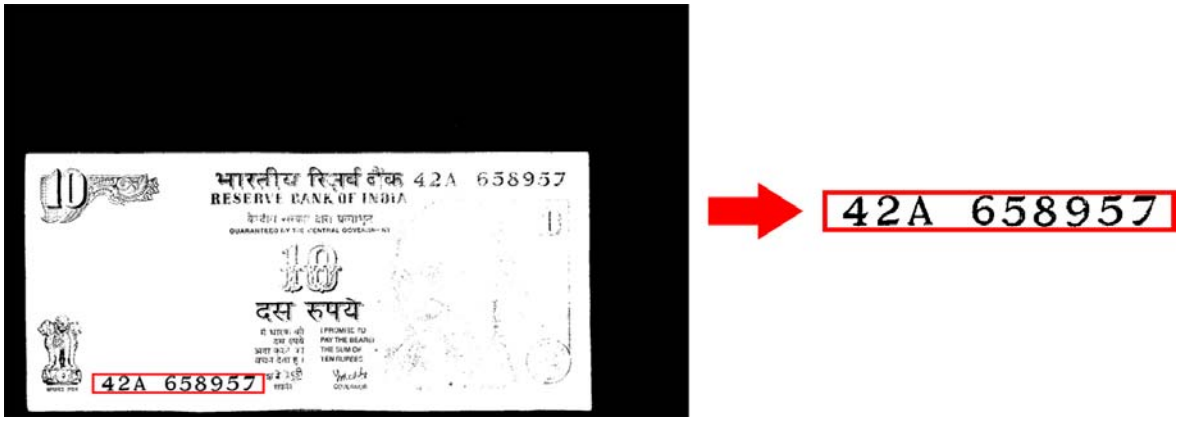


Abbildung 21

Als Vorbereitung für die Methode „getBlackPixelFromTopWithRed“, welche für die Breite und X-Koordinate der Seriennummer zuständig ist, wird ebenfalls ein Bildausschnitt vom gefilterten Gesamtbild erzeugt. Dieser Bildausschnitt entspricht aber nur der Breite der Region der jeweiligen Nummer, die im Testdeck angegeben ist. Die Höhe des Bildausschnittes und die Y-Koordinate wird aus dem Array genommen, welches von der vorherigen Methode gefüllt wurde.

```objc
- (id) getBlackPixelFromTopWithRed:(UInt8)red withBlue:(UInt8)blue withGreen:(UInt8)green
                         withAlpha:(UInt8)alpha withImage:(UIImage*)image2
                     andBlackAreas:(int)blackAreas andWidth:(int) snrWidth
{
    CFDataRef pixelData = CGDataProviderCopyData(CGImageGetDataProvider(image2.CGImage));
    const UInt8* data = (UInt8 *) CFDataGetBytePtr(pixelData);
    int imageWidth = (int)CGImageGetWidth(image2.CGImage);
    int imageHeight = (int)CGImageGetHeight(image2.CGImage);
    int pixelX = imageWidth-1; // The X coordinate of the pixel you want to retrieve
    int pixelY = imageHeight-1; // The Y coordinate of the pixel you want to retrieve
    int blackPixelPerCol = 0;
    int blackCounter = 0;
    int whiteCounter = 0;
    money = [[billDatas alloc] init];
    NSMutableArray *blackWhiteCols = [[NSMutableArray alloc] init];
    NSMutableArray *Position = [[NSMutableArray alloc] init];
    NSMutableArray *swSyntax = [[NSMutableArray alloc] init];
    NSMutableArray *BlackArray = [[NSMutableArray alloc] init];
    NSMutableArray *WhiteArray = [[NSMutableArray alloc] init];
    NSMutableArray *StartPositionBlackArray = [[NSMutableArray alloc] init];
    UInt8 _red,_green,_blue,_alpha;

    for (int i=0; i<pixelX; i++) {
        int counter=0;
        while (counter < pixelY) {
            int pixelInfo = ((imageWidth  * counter) + i ) * 4;
            if ([[[UIDevice currentDevice] model] isEqualToString:@"iPad Simulator"]) {
                _red   = data[pixelInfo + 0];
                _green = data[pixelInfo + 1];
                _blue  = data[pixelInfo + 2];
                _alpha = data[pixelInfo + 3];
            }else{
                _blue  = data[pixelInfo + 0];
                _green = data[pixelInfo + 1];
                _red   = data[pixelInfo + 2];
                _alpha = data[pixelInfo + 3];
            }
            if (_red == red && _green == green && _blue == blue && _alpha == alpha) {
                blackPixelPerCol++;
            }
            counter++;
        }
        if (blackPixelPerCol >= 2) {
            if (whiteCounter != 0) {
                [WhiteArray addObject:[NSNumber numberWithInt:whiteCounter]];
            }
            whiteCounter = 0;
            [blackWhiteCols addObject:@"s"];
            blackCounter++;
            if (i==pixelX-1 && blackCounter != 0 && blackCounter > 10) {
                [StartPositionBlackArray addObject:[NSNumber numberWithInt:(i-blackCounter)]];
            }
        }else{
            if (blackCounter != 0 && blackCounter > 10) {
                [BlackArray addObject:[NSNumber numberWithInt:blackCounter]];
                [StartPositionBlackArray addObject:[NSNumber numberWithInt:(i-blackCounter)]];
            }
            blackCounter = 0;
            [blackWhiteCols addObject:@"w"];
            whiteCounter++;
        }
        blackPixelPerCol = 0;
    }
    NSString *lastCharacter = @"";
    for (NSString *Zeichen in blackWhiteCols) {
        if (![Zeichen isEqualToString:lastCharacter]){
            [swSyntax addObject:Zeichen];
        }
        lastCharacter = Zeichen;
    }
    if (whiteCounter != 0) {
        [WhiteArray addObject:[NSNumber numberWithInt:whiteCounter]];
    }
    if (blackCounter != 0 && blackCounter > 12 && blackCounter <= 40) {
        [BlackArray addObject:[NSNumber numberWithInt:blackCounter]];
    }
    if ([BlackArray count] == blackAreas && [WhiteArray count] >= (blackAreas-1)) {
        int x = [[StartPositionBlackArray objectAtIndex:0] intValue] - 8;
        int width = ([[StartPositionBlackArray lastObject] intValue]+[[BlackArray lastObject] intValue]-
                     [[StartPositionBlackArray objectAtIndex:0] intValue] + 16);
        [Position addObject:[NSNumber numberWithInt:x]];
        [Position addObject:[NSNumber numberWithInt:width]];
    }
    else if ([BlackArray count] < blackAreas){
        [Position addObject:[NSNumber numberWithInt:0]];
        [Position addObject:[NSNumber numberWithInt:imageWidth]];
    }
    else if ([BlackArray count] > blackAreas){
        int numberWidth = 0;
        int startWhite = 0;
        if ([[swSyntax objectAtIndex:0] isEqualToString:@"w"]) {
            startWhite = 1;
        }
        for (int j = 0; j <= ([BlackArray count] - blackAreas); j++) {
            for (int i = 0; i < blackAreas; i++) {
                numberWidth += [[BlackArray objectAtIndex:i] intValue];
            }
            for (int i = 0; i < (blackAreas-1); i++) {
                if (i < [WhiteArray count]) {
                    numberWidth += [[WhiteArray objectAtIndex:(startWhite+i)] intValue];
                }

            }
            if (numberWidth >= (snrWidth-pointToMM) && numberWidth <= (snrWidth+pointToMM)){
                [Position addObject:[NSNumber numberWithInt:([[StartPositionBlackArray objectAtIndex:j] intValue]-5)]];
                [Position addObject:[NSNumber numberWithInt:numberWidth]];
            }
            numberWidth = 0;
        }
    }
    CFRelease(pixelData);
    return Position;
}
```

Abbildung 22

Die Methode „getBlackPixelFromTopWithRed“ wird ebenfalls im Zustand „WaitOnPicture“ des Hauptautomaten nach der Methode „getBlackPixelFromLeftWithRed“ aufgerufen. Dabei werden Parameter für die RGB-Werte und den Alpha-Wert übergeben. Außerdem wird der Bildausschnitt, in dem sich die Seriennummer befindet, mitgeliefert. Zudem wird noch die Anzahl der Zeichen („blackAreas“) und die Breite der Seriennummer („snrWidth“) aus dem Testdeck an die Methode übergeben.

Bei dieser Methode gibt es jedoch noch weitere Unterschiede zur „getBlackPixelFrom-LeftWithRed“-Methode. Da die einzelnen Zeichen mit Weißbereichen voneinander getrennt sind, muss jeder Character einzeln gespeichert werden und im späteren Verlauf zu einem gesamten Bereich zusammengefügt werden.

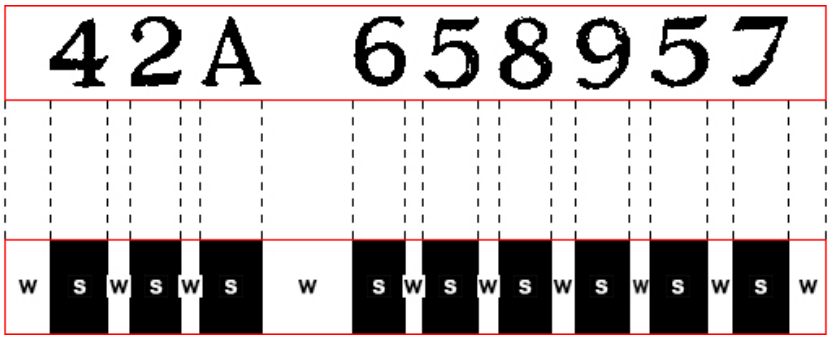


Abbildung 23

Es müssen alle Weiß- und Schwarzbereiche in einem Array abgespeichert werden, um im Nachhinein die X-Koordinate und Breite der Seriennummer aus den einzelnen Bereichen herleiten zu können.

Auch zu Beginn dieser Methode wird der Bildausschnitt mit Hilfe von „CFDataGetBytePtr“ in einen UInt8 umgewandelt und für das spaltenweise auslesen vorbereitet.

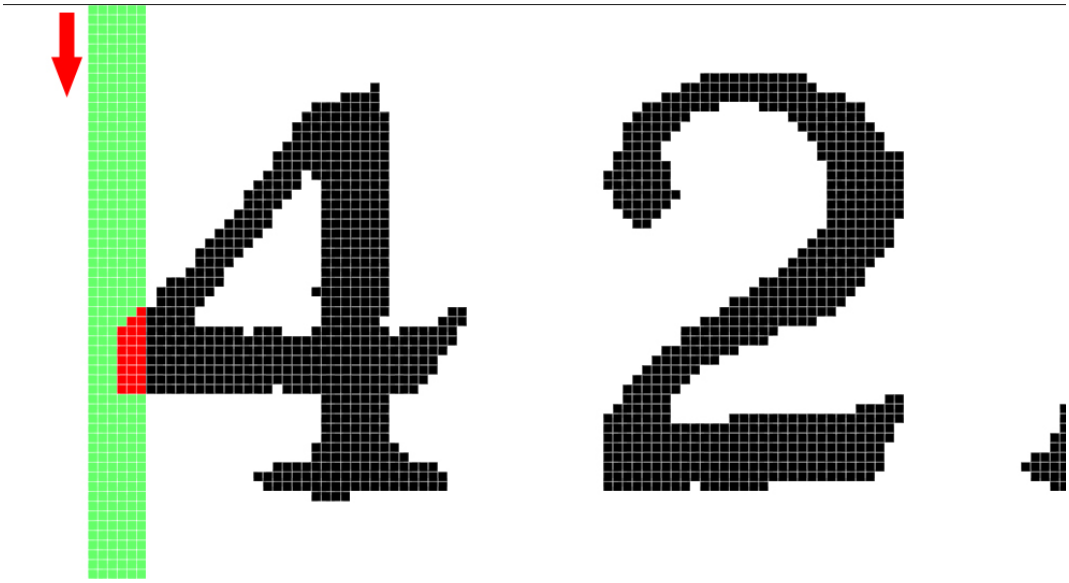


Abbildung 24

Innerhalb einer „for“-Schleife werden alle Pixel spaltenweise, links beginnend, von der oberen Bildkante aus geprüft. Entsprechen die drei Farbwerte gleich null und der Alphawert ist gleich 1, so ist das Pixel schwarz. Für jedes schwarze Pixel in der Spalte wird der Counter „blackPixelPerCol“ um 1 hochgezählt. Wurde eine Zeile komplett durchlaufen, wird dieser Counter geprüft. Sind in der Zeile 2 oder mehr schwarze Pixel enthalten, so ist es wahrscheinlich, dass diese Spalte zur Seriennummer gehört, ansonsten wird diese Spalte als Weißbereich anerkannt.

Enthält die Spalte nun 2 oder mehr schwarze Punkte, so wird geprüft, ob vorher bereits weiße Spalten gefunden wurden. Die Anzahl der weißen Spalten befindet sich im „whiteCounter“. Sollten vorher weiße Spalten gefunden worden sein, so wird der Wert des Counters dem „WhiteArray“ hinzugefügt. Damit wurde ein Weißbereich gefunden.

Unabhängig vom „whiteCounter“ wird dieser beim Auslesen einer Spalte mit 2 oder mehr schwarzer Pixel gleich null gesetzt. Das Array „blackWhiteCols“, welches einen Wert („s“ für schwarze Spalte, „w“ für weiße Spalte) für jede Spalte des Bildes speichert, bekommt den Wert „s“ zugewiesen und der „blackCounter“, welcher die schwarzen Spalten zählt, wird um 1 erhöht. Sollte der letzte Schwarzbereich bis zur rechten Kante des Bildes reichen, so wird die Startspalte dieses Bereiches in das Array „StartPositionBlackArray“ gespeichert. Dieses Array beinhaltet die Startspalten aller Schwarzbereiche.

Sollte keine schwarze Spalte gefunden werden, wird zuerst geprüft, ob der „blackCounter“ einen Wert ungleich 0 besitzt und größer 10 ist. In diesem Fall wird die Anzahl der Spalten des Schwarzbereiches in das Array „BlackArray“ gespeichert. Außerdem wird die Startposition des Bereiches in „StartPositionBlackArray“ geschrieben.

Unabhängig davon, ob vorher bereits ein Schwarzbereich gefunden wurde, wird der „blackCounter“ gleich null gesetzt. Das „blackWhiteCols“-Array erhält den Wert „w“ für die Spalte und der „whiteCounter“ zählt um 1 hoch.

Ist das spaltenweise Auslesen des Bildes beendet, wird in einer „for“-Schleife das Array „swSyntax“ mit Werten gefüllt, wobei der erste Wert dem ersten Eintrag von „blackWhiteCols“ entspricht und der zweite Wert entspricht dem ersten Eintrag des „blackWhiteCols“-Arrays, der einen anderen Wert besitzt, wie die Einträge davor. Somit entsteht ein Array mit den sich abwechselnden Einträgen „s“ und „w“ für die Anzahl der Schwarz- und Weißbereiche.

Da die Spaltenanzahl des letzten Bereiches noch nicht gespeichert ist, wird geprüft ob der „whiteCounter“ oder der „blackCounter“ ungleich null ist. Im jeweiligen Fall wird der Wert des Counters als Bereichsbreite abgespeichert.

Dank der Übergabe der Anzahl der Zeichen, aus welchen die Seriennummer besteht, kann jetzt überprüft werden, ob auch so viele Schwarzbereiche gefunden wurden. Außerdem muss zwischen jedem Zeichen ein Weißbereich liegen. Dies lässt sich auch gut überprüfen, in dem man abfragt, ob die Anzahl der Weißbereiche größer gleich der An-

zahl der Zeichen minus 1 ist. In diesem Fall wird als X-Koordinate die Anfangsspalte des ersten Schwarzbereiches verwendet und ein Puffer von 8 Pixel abgezogen. Die Breite der Seriennummer setzt sich dann wie folgt zusammen. Es wird die Anfangsspalte des letzten Schwarzbereiches mit der Breite dieses Bereiches addiert und davon die Spaltennummer der Anfangsspalte des ersten Schwarzbereiches subtrahiert. Auch dabei wird ein Puffer von 8 Pixel gewährt. Diese beiden Werte werden anschließend in das Array „Position“ geschrieben und dieses wird am Ende der Methode zurückgeliefert.

Sollte die Anzahl der Schwarzbereiche kleiner der Anzahl der Zeichen sein, wird die X-Koordinate auf 0 gesetzt und die Breite erhält den Wert der Breite des Bildausschnittes.

Wurden mehr Schwarzbereiche wie die Anzahl der Zeichen gefunden, dann werden immer so viele aneinander liegende Schwarzbereiche wie „blackAreas“ für die Breite addiert. Außerdem werden die Weißbereiche, welche zwischen diesen Schwarzbereichen liegen, ebenfalls zu der Breite dazu addiert. Entspricht die Breite nun der aus dem Testdeck an die Methode mitgelieferten Breite, so wird diese und die Y-Koordinate dieser Variante in das Array „Position“ gespeichert. Wenn nicht wird dies für alle anderen Möglichkeiten wiederholt.

4.4 Aufbereiten der Scan-Positionen

Nachdem jetzt versucht wurde den genauen Bereich der Seriennummer zu finden, muss die Scan-Position für die OCR vorbereitet werden. Dabei kann es vorkommen, dass genau eine oder auch sehr viele Positionen vorhanden sind, in denen sich die Seriennummer befindet. Um dies Auszuwerten wurde eine Methode „getScanPositionRects“ erstellt.

```
- (void) getScanPositionRects{
    firstPosRects = [[NSMutableArray alloc] init];
    secondPosRects = [[NSMutableArray alloc] init];
    int xf, yf, wf, hf = 0;
    int xs, ys, ws, hs = 0;

    if ([firstNumberPosition count] == 0 || [blackPixelArrayFirst count] == 0) {
        firstPosRects = [money firstPosRects];
    }
    else{
        for (int i=0; i<[firstNumberPosition count]; i+=2) {
            yf = [[blackPixelArrayFirst objectAtIndex:0] intValue]+[[[money firstBigRegion] objectAtIndex:1] intValue];
            hf = [[blackPixelArrayFirst objectAtIndex:1] intValue];
            xf = [[firstNumberPosition objectAtIndex:i] intValue]+[[[money firstBigRegion] objectAtIndex:0] intValue];
            wf = [[firstNumberPosition objectAtIndex:(i+1)] intValue];
            [firstPosRects addObject:[NSValue valueWithCGRect:CGRectMake(xf,yf,wf,hf)]];
        }
    }

    if ([[numberCountArray objectAtIndex:0] isEqualToString:@"two"]) {
        if ([secondNumberPosition count] == 0 || [blackPixelArraySecond count] == 0) {
            secondPosRects = [money secondPosRects];
        }
        else{
            for (int i=0; i<[secondNumberPosition count]; i+=2) {
                ys = [[blackPixelArraySecond objectAtIndex:0] intValue]+
                    [[[money secondBigRegion] objectAtIndex:1] intValue];

                hs = [[blackPixelArraySecond objectAtIndex:1] intValue];

                xs = [[secondNumberPosition objectAtIndex:i] intValue]+
                    [[[money secondBigRegion] objectAtIndex:0] intValue];

                ws = [[secondNumberPosition objectAtIndex:(i+1)] intValue];

                [secondPosRects addObject:[NSValue valueWithCGRect:CGRectMake(xs,ys,ws,hs)]];
            }
        }
    }
}
```

Abbildung 25

Diese Methode prüft die Scan-Positionen für alle Seriennummern, die auf der Banknote vorhanden sind, einzeln und stellt diese in einem Array für die weitere Verarbeitung zur Verfügung.

Zu Beginn wird überprüft, ob die Arrays „firstNumberPosition“ oder „blackPixelArrayFirst“ leer sind. In „firstNumberPosition“ sollten, nach erfolgreichem Ermitteln der X-Koordinate und der Breite, Werte für diese beiden Parameter der ersten Seriennummer enthalten sein. Das Array „blackPixelArrayFirst“ müsste, wenn gefunden, die Werte für Y-Koordinate und Höhe der ersten Nummer beinhalten. Sollte eines dieser Arrays nicht gefüllt sein, werden die Positionen für das Scannen aus dem Testdeck entnommen. Dabei wird die große Region verwendet und immer um eine Toleranz mit einer Schrittgröße verschoben.

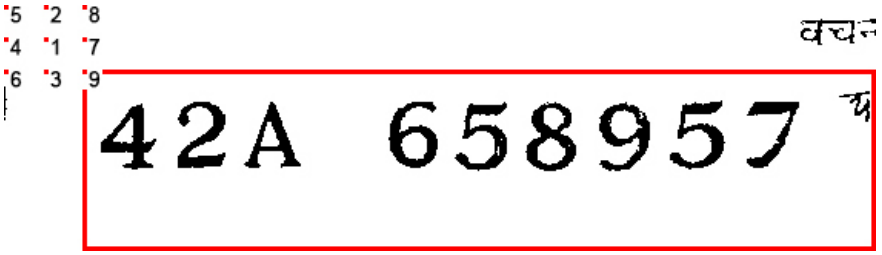


Abbildung 26

In diesem Fall kann es zu vielen Scan-Positionen kommen, da dies abhängig von der Toleranz und Schrittgröße ist. Sollte dies aber eintreten, so werden die Positionen geordnet. So wie in der Abbildung 26 zu sehen, befindet sich die erste Position im Zentrum, da sie auch den Koordinaten der Region entspricht, welche im Testdeck angegeben ist. Nachfolgend werden zuerst alle Positionen in y-Richtung schrittweise abgearbeitet. Danach wird in x-Richtung um eins nach links verschoben und da wieder alle Positionen in y-Richtung durchlaufen. Ist das abgeschlossen, wird die Position, von der Startposition aus, um ein nach rechts verschoben und auch da wieder alle Möglichkeiten abgearbeitet.

Sind die beiden Arrays aber mit Werten gefüllt, so werden andere Scan-Positionen ermittelt. In einer „for“-Schleife wird das Array „firstPosRects“ mit Werten gefüllt. Allerdings ist es nur möglich, dass mehrere Werte für Breite und X-Koordinate vorhanden sind. Daher wird nur im „firstNumberPosition“-Array nach mehreren Werten gesucht. Für jeden Durchlauf der Schleife, werden die Werte als CGRect, welches aus X-Koordinate, Y-Koordinate, Breite und Höhe besteht, in das Array „firstPosRects“ gespeichert.

Abbildung 27

Dabei kann es vorkommen, dass mehrere Positionen zum Scannen, wie in der Abbildung 27 eingezeichnet, möglich sind. Die rot eingezeichnete Position wäre die optimale, aber die blau Eingezeichnete könnte ebenfalls möglich sein, da die Anzahl der Schwarz- und

Weißbereiche korrekt ist und die Breite im Toleranzbereich der vorgegebenen Breite ist. Hier wird aber ebenfalls zuerst die weiter links ermittelte Position ausgelesen. Anschließend folgt die blaue Scan-Position.

Natürlich kann es auch möglich sein, dass im „firstNumberPosition“-Array nur ein Wert für die X-Koordinate und Breite vorhanden ist. Dies wäre natürlich der Optimalfall, denn somit sollte die Position der Seriennummer genau ermittelt worden sein.

Abbildung 28

Wenn dies der Fall ist, so sollte die OCR das bestmögliche Ergebnis zurückliefern. Alle störenden schwarzen Pixel, welche nicht zur Seriennummer gehören, werden somit nicht betrachtet und führen zu keinen falschen Ergebnissen.

Sollte nun auf der Banknote noch eine zweite Seriennummer vorhanden sein, wird das gleiche Verfahren für das Ermitteln der Scan-Positionen auf diese Nummer angewendet. Dabei werden die Arrays „blackPixelArray“ (Werte für Y-Koordinate und Höhe) und „secondNumberPosition“ (Werte für X-Koordinate und Breite) verwendet um die Positionen zu erzeugen. Die möglichen Scan-Positionen werden in das Array „secondPosRects“ geschrieben und für den weiteren Ablauf zur Verfügung gestellt.

4.5 Setzen der Filterparameter & Positionen für den jeweiligen Durchlauf

Um alles für die OCR vorzubereiten, wurde eine Methode „filterForOCR“ angelegt. In dieser Methode werden neben den Scan-Positionen auch die Filterparameter für den jeweiligen Durchgang der Texterkennung definiert und bereitgestellt.

```
-(void)filterForOCR:(UIImage *)image position:(int)z{
    if (z==0) {
        if (durchlauf < ([firstPosRects count]*[[money FirstThresholdMultiplier] count])) {
            scanPossition = [[firstPosRects objectAtIndex:firstRectPosCounter] CGRectValue];
            if (filterDurchlaufFirst < [[money FirstThresholdMultiplier] count]) {
                if ([[[money filterColorArray1] objectAtIndex:0] isEqualToString:@"all"]) {
                    bwImageFil = [filtertImages getImageFromColorMatrixForColor:image
                                                withRed:[[[money filterColorValueArray1] objectAtIndex:0] floatValue]
                                                withGreen:[[[money filterColorValueArray1] objectAtIndex:1] floatValue]
                                                withBlue:[[[money filterColorValueArray1] objectAtIndex:2] floatValue]];
                }else{
                    bwImageFil = [filtertImages getImageFromColorMatrixForGray:image
                                                withRed:[[[money filterColorValueArray1] objectAtIndex:0] floatValue]
                                                withGreen:[[[money filterColorValueArray1] objectAtIndex:1] floatValue]
                                                withBlue:[[[money filterColorValueArray1] objectAtIndex:2] floatValue]];
                }
                bwImageFil = [filtertImages getAverageLuminanceThresholdFilterImage:bwImageFil
                                                withThresholdMultiplier:[[[money FirstThresholdMultiplier]
                                                                                objectAtIndex:filterDurchlaufFirst] floatValue]];

                filterDurchlaufFirst++;
                _activityIndicator.hidden = false;
                [self.activityIndicator startAnimating];
                self.imageToRecognize.image = [filtertImages GetImageByDrawingCircleOnImage:image
                                                                        withCGRect:scanPossition];
                [self performSelectorOnMainThread:@selector(makeMyProgressBarMoving) withObject:nil waitUntilDone:NO];
            }
            else{
                firstRectPosCounter++;
                filterDurchlaufFirst = 0;
            }
        }
    }
    else if (z==1){
        if (durchlauf < ([secondPosRects count]*[[money SecondThresholdMultiplier] count])) {
            scanPossition = [[secondPosRects objectAtIndex:secondRectPosCounter] CGRectValue];
            if (filterDurchlaufSecond < [[money SecondThresholdMultiplier] count]) {
                if ([[[money filterColorArray2] objectAtIndex:0] isEqualToString:@"all"]) {
                    bwImageFil = [filtertImages getImageFromColorMatrixForColor:image
                                                withRed:[[[money filterColorValueArray2] objectAtIndex:0] floatValue]
                                                withGreen:[[[money filterColorValueArray2] objectAtIndex:1] floatValue]
                                                withBlue:[[[money filterColorValueArray2] objectAtIndex:2] floatValue]];
                }else{
                    bwImageFil = [filtertImages getImageFromColorMatrixForGray:image
                                                withRed:[[[money filterColorValueArray2] objectAtIndex:0] floatValue]
                                                withGreen:[[[money filterColorValueArray2] objectAtIndex:1] floatValue]
                                                withBlue:[[[money filterColorValueArray2] objectAtIndex:2] floatValue]];
                }
                bwImageFil = [filtertImages getAverageLuminanceThresholdFilterImage:bwImageFil
                                                withThresholdMultiplier:[[[money SecondThresholdMultiplier]
                                                                                objectAtIndex:filterDurchlaufSecond] floatValue]];

                filterDurchlaufSecond++;
                self.imageToRecognize.image = [filtertImages GetImageByDrawingCircleOnImage:image
                                                                        withCGRect:scanPossition];
                [self performSelectorOnMainThread:@selector(makeMyProgressBarMoving) withObject:nil waitUntilDone:NO];
            }
            else{
                secondRectPosCounter++;
                filterDurchlaufSecond = 0;
            }
        }
    }
}
```

Abbildung 29

Wenn die Methode aufgerufen wird, muss ein Parameter für die jeweilige Nummer übergeben werden. Dabei stehen der Parameter „0“ für die erste Seriennummer auf der Banknote und der Parameter „1“ für die Zweite. Wird nun die „0“ übergeben, wird der Counter „durchlauf“, welcher immer nach Überprüfung der Seriennummern hochgezählt wird, mit der Anzahl der Scan-Positionen mal der Anzahl der Filterparameter verglichen. Ist die Variable „durchlauf“ kleiner der Anzahl dieser beiden Parameter, so wird zuerst die Scan-Position ermittelt. Dabei wird der Wert aus dem „firstPosRects“-Array an der Stelle „firstRectPosCounter“, welche zu Beginn 0 entspricht und um 1 erhöht wird, wenn nach allen Filterparametern an dieser Position kein korrektes Ergebnis gefunden wurde, entnommen. Dieses Array wurde vorher in der „getScanPositionRects“-Methode befüllt.

Danach wird mittels einer Vergleichsvariable „filterDurchlaufFirst“, welche zu Beginn 0 entspricht, überprüft ob alle Filterparameter verwendet wurden. Sollte also „filterDurchlaufFirst“ kleiner der Anzahl der Filterparameter sein, wird der Filterparameter verändert.

Dabei wird überprüft, welcher Farbfilter im Testdeck angelegt wurde. Beinhaltet dieser einen R-, G- und B-Wert, so werden alle drei Farben verwendet und es entsteht kein Graubild. Somit wird ein UIImage „bwImageFil“ angelegt, welches von der Methode „getImageFromColorMatrixForColor“ erzeugt wird. Dabei wird das Ausgangsbild an die Methode übergeben. Ist jedoch nur ein R-, G- oder B-Wert im Testdeck angegeben, so entsteht ein Graubild, welches von der Methode „getImageFromColorMatrixForGray“ aus dem Ausgangsbild erzeugt wurde.

Abbildung 30

Dieses „bwImageFil“ wird verwendet um mittels der Methode „getAverageLuminanceThresholdFilterImage“ ein für die OCR fertig gefiltertes Bild zu erzeugen.

Anschließend wird die Vergleichsvariable „filterDurchlaufFirst“ um 1 hochgezählt und ein neues Bild mit einem Rahmen an der Stelle der Scan-Position in der ImageView ausgegeben.

Sollten jedoch alle Filterparameter verwendet worden sein und es wurde keine korrekte Seriennummer ausgelesen, so wird die Scan-Position, wenn möglich, verändert (der nächste Wert aus dem Array wird verwendet) und „filterDurchlaufFirst“ wird wieder auf 0 gesetzt.

Ist eine zweite Seriennummer auf der Banknote enthalten, wird die Methode „filterForOCR“ mit dem Parameter 1 aufgerufen. Für diese Nummer gilt der gleiche Ablauf nur mit den, für diese Nummer spezifischen, Scan-Positionen und Filterparametern.

4.6 Syntax-Check

Um die, von der OCR gelieferten, Ergebnisse auf Korrektheit zu überprüfen, wird eine Überprüfung des Syntax vorgenommen. Dabei wird die Angabe der Syntax aus dem jeweiligen Testdeck verwendet und die einzelnen Zeichen, welche von der Texterkennung ermittelt wurden, werden mit dieser Syntax verglichen.

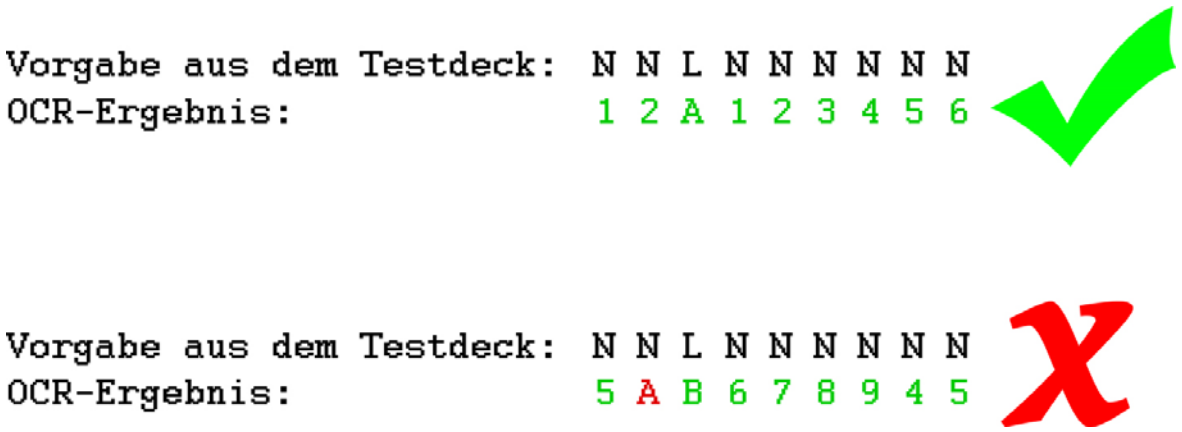


Abbildung 31

Im Testdeck ist der Syntax als String definiert, welcher aus den Buchstaben N, L, B und S für „Number“, „Letter“, „Both“ und „Space“ besteht. Sollten die ausgelesenen Character mit der Vorgabe übereinstimmen, so ist der ermittelte String in Hinsicht auf Syntax korrekt. Im Fall von einer oder mehr Abweichungen, gilt die ermittelte Seriennummer allerdings als unkorrekt und wird direkt verworfen.

Um diesen Ablauf programmiertechnisch umzusetzen, wurde die Methode „syntaxCheck“ erstellt und in den Ablauf des Hauptautomaten integriert.

```
-(BOOL)syntaxCheck:(NSString*) recognizedString andNumberCount:(NSString*) position{
    NSMutableArray *recognizedArray = [[NSMutableArray alloc] init];
    recognizedString = [recognizedString stringByTrimmingCharactersInSet:[NSCharacterSet newlineCharacterSet]];
    recognizedString = [recognizedString stringByReplacingOccurrencesOfString:@" " withString:@""];
    for (int i = 0; i < [recognizedString length]; i++) {
        NSString *ch = [recognizedString substringWithRange:NSMakeRange(i, 1)];
        [recognizedArray addObject:ch];
    }
    if ([position isEqualToString:@"first"]) {
        if ([recognizedArray count]!=firstNumberObject.syntaxLength) {
            if([recognizedArray count]>firstNumberObject.syntaxLength){
                recognizedString = [self percentCheck:percentArray withStringArray:recognizedArray];
                [recognizedArray removeAllObjects];
                for (int i = 0; i < [recognizedString length]; i++) {
                    NSString *ch = [recognizedString substringWithRange:NSMakeRange(i, 1)];
                    [recognizedArray addObject:ch];
                }
                if ([recognizedArray count]!=firstNumberObject.syntaxLength) {
                    return false;
                }
            }else{
                return false;
            }
        }
        for (int i = 0; i<[recognizedArray count]; i++) {
            const unichar uch = [recognizedString characterAtIndex:i];
            if ([[firstNumberObject.syntaxArrayWithoutSpaces objectAtIndex:i] isEqualToString:@"N"]) {
                if (!(0x0030 <= uch && uch <= 0x0039)) {
                    return false;
                }
            }
            else if ([[firstNumberObject.syntaxArrayWithoutSpaces objectAtIndex:i] isEqualToString:@"L"]){

                if ((0x0030 <= uch) && (uch <= 0x0039)){
                    return false;
                }
            }
        }
    }
    else if([position isEqualToString:@"second"]){
        if ([recognizedArray count]!=secondNumberObject.syntaxLength) {
            if([recognizedArray count]>secondNumberObject.syntaxLength){
                recognizedString = [self percentCheck:percentArray withStringArray:recognizedArray];
                [recognizedArray removeAllObjects];
                for (int i = 0; i < [recognizedString length]; i++) {
                    NSString *ch = [recognizedString substringWithRange:NSMakeRange(i, 1)];
                    [recognizedArray addObject:ch];
                }
                if ([recognizedArray count]!=secondNumberObject.syntaxLength) {
                    return false;
                }
            }else{
                return false;
            }
        }
        for (int i = 0; i<[recognizedArray count]; i++) {
            const unichar uch = [recognizedString characterAtIndex:i];
            if ([[secondNumberObject.syntaxArrayWithoutSpaces objectAtIndex:i] isEqualToString:@"N"]) {
                if (!(0x0030 <= uch && uch <= 0x0039)) {
                    return false;
                }
            }
            else if ([[secondNumberObject.syntaxArrayWithoutSpaces objectAtIndex:i] isEqualToString:@"L"]){

                if ((0x0030 <= uch) && (uch <= 0x0039)){
                    return false;
                }
            }
        }
    }
    int recognizedBlockCounter = 0;
    for (G8RecognizedBlock *recognizedBlock in confidence) {
        if (([recognizedString length] > recognizedBlockCounter) &&
            [[recognizedString substringWithRange:(NSRange){recognizedBlockCounter,1}]
             isEqualToString:[recognizedBlock text]])
        {
            if ([buildedStringArray count] == recognizedBlockCounter) {
                [buildedStringArray addObject:[recognizedBlock text]];
                [buildedStringArrayPercents addObject:[NSNumber numberWithFloat:[recognizedBlock confidence]]];
            }else{
                if ([[buildedStringArray objectAtIndex:recognizedBlockCounter]
                     isEqualToString:[recognizedBlock text]])
                {
                    if ([recognizedBlock confidence] >
                        [[buildedStringArrayPercents objectAtIndex:recognizedBlockCounter] floatValue])
                    {
                        [buildedStringArrayPercents replaceObjectAtIndex:recognizedBlockCounter
                                                              withObject:[NSNumber numberWithFloat:[recognizedBlock confidence]]];
                    }
                }else{
                    if ([recognizedBlock confidence] >
                        [[buildedStringArrayPercents objectAtIndex:recognizedBlockCounter] floatValue])
                    {
                        [buildedStringArrayPercents replaceObjectAtIndex:recognizedBlockCounter
                                                              withObject:[NSNumber numberWithFloat:[recognizedBlock confidence]]];
                        [buildedStringArray replaceObjectAtIndex:recognizedBlockCounter
                                                      withObject:[recognizedBlock text]];
                    }
                }
            }
            recognizedBlockCounter++;
        }
    }
    buildedString = [buildedStringArray componentsJoinedByString:@""];
    recognizedText = [recognizedString copy];
    return true;
}
```

Abbildung 32

Zu Beginn dieser Methode werden alle Leerzeichen und Zeilenumbrüche aus dem OCR-Ergebnis entfernt um einen reinen Textstring zu erhalten. Anschließend wird in einer „for“-Schleife der String in seine einzelnen Zeichen aufgetrennt und diese in ein Array gespeichert. Danach wird überprüft, um welche Seriennummer es sich handelt, da auch die Möglichkeit besteht, dass zwei Seriennummern mit unterschiedlicher Syntax auf der Banknote vorhanden sein können.

Für die jeweilige Position wird die Information zum Syntax aus dem Testdeck abgerufen. Zuerst wird die vorgegebene Länge der Seriennummer verglichen. Diese ergibt sich aus der Anzahl der Zeichen, welche im Testdeck als Syntax angegeben wurden. Sollte die Länge der Seriennummer, die von der OCR ermittelt wurde nicht mit der korrekten Länge übereinstimmen, wird überprüft, ob sie kürzer oder länger ist. Ist sie kürzer, so wird diese Seriennummer direkt verworfen. Sollte die ermittelte Seriennummer länger als die Vorgabe sein, so wird die Methode „percentCheck“ aufgerufen, welche einen gekürzten String zurückliefert. Anschließend wird der Inhalt des „recognizedArray“ gelöscht und das Array wird mit den einzelnen Charakteren des neuen Strings befüllt. Entspricht nun die Länge des neuen Strings nicht der im Testdeck angegebenen Länge, wird diese Nummer ebenfalls verworfen.

Nach dem Vergleich der Länge des Strings, wird die Einhaltung der Syntaxvorgabe überprüft. Dazu wird wieder in einer „for“-Schleife jedes einzelne Zeichen des, von der Länge korrekten, Strings in einen „unichar“ gewandelt. Anschließend wird die Syntaxvorgabe aus dem Testdeck verwendet, um zu überprüfen, ob ein Buchstabe oder eine Zahl an der jeweiligen Stelle stehen muss. Sollte an der aktuellen Stelle ein „N“ (für Number) stehen, wird geprüft, ob der „unichar“ in der ASCII-Tabelle zwischen den Werten 30 und 39 liegt, da diese Werte alle Zahlen von 0 bis 9 enthalten. Ist dies nicht so, wird die Nummer verworfen, ansonsten wird mit der nächsten Stelle des Strings fortgesetzt. Das gleiche Prinzip wird angewendet, wenn an der aktuellen Stelle ein „L“ (für Letter) steht. Jedoch wird dann überprüft, ob es sich nicht um eine Zahl handelt. Sollte also der „unichar“ zwischen den Werten 30 und 39 der ASCII-Tabelle liegen entspricht er einer Zahl und ist somit falsch. In diesem Fall wird wieder abgebrochen. Für den Fall, dass ein „B“ in der Syntax enthalten ist kann beides (Zahl oder Buchstabe) vorkommen. Ansonsten wird mit der nächsten Stelle des Strings fortgefahren.

Die gleiche Vorgehensweise wird auf eine zweite, möglicherweise vorhandene, Seriennummer angewendet. Sollten Seriennummern ausgelesen wurden, welche der Länge und der Syntax der Vorgabe aus dem Testdeck entsprechen, werden diese im weiteren Verlauf verwendet.

Anschließend werden die einzelnen Zeichen der Strings mit ihren zugehörigen Prozentwerten in einer „for“-Schleife ausgewertet. Dazu wird ein Counter angelegt, welcher nach jeden Durchlauf der Schleife um eins erhöht wird. Dieser „recognizedBlockCounter“ wird allerdings immer vor Beginn der Schleife auf 0 gesetzt.

Als Erstes wird überprüft, ob die Länge des Strings größer ist als der Wert des Counters und ob das Zeichen des Strings an der Stelle des Counters gleich dem ausgelesenen Zeichen ist. In diesem Fall wird weiter geprüft, ob die Größe des „buildedStringArray“ gleich dem Wert des Counters ist. Trifft dies zu wird das Zeichen in dem Array gespeichert und der dazugehörige Prozentwert der OCR wird in einem anderen Array „buildedStringArrayPercents“ gesichert. Sollte die Größe des „buildedStringArray“ nicht gleich dem Wert des Counters sein, wird geprüft, ob der Inhalt des „buildedStringArray“ an der Stelle des Counters gleich dem aktuell ausgelesenem Zeichen an der jeweiligen Stelle entspricht. Ist dies der Fall, werden die Prozentwerte verglichen und der höhere Prozentwert wird im „buildedStringArrayPercents“ gesichert. Steht allerdings ein anderes Zeichen an dieser Stelle, wird anhand der Prozentwerte überprüft, welches Korrekt ist. So wird entweder der Wert im Array überschrieben oder der neue Wert, aufgrund einer zu niedrigen Prozentzahl verworfen.

Nach dem Befüllen der beiden Arrays, wird aus dem „buildedStringArray“ ein String gebildet, welcher als lokale Variable allen Methoden der Klasse zur Verfügung steht und später bei dem Vergleich der Seriennummern zum Einsatz kommt.

4.7 Prozent-Check

Wie in der kurzen Erläuterung der Methode „syntaxCheck“ beschrieben, wird eine Methode benötigt, welche die Seriennummer mittels der OCR-Prozentwerte ändern kann. Dies ist natürlich nur sinnvoll, wenn die ausgelesene Nummer mehr Zeichen enthält als die Vorgabe im Testdeck. Sollte die ermittelte Nummer weniger Zeichen enthalten, müssten neue Zeichen hinzugefügt werden, welche nirgendwo definiert sind und somit frei erfunden wären.

Die Methode „percentCheck“ beherrscht diese Funktionen und liefert einen veränderten String zurück.

```
1005 -(NSString*) percentCheck:(NSMutableArray*) singlePercentArray
1006            withStringArray:(NSMutableArray*) oldRecognizedStringArray
1007 {
1008     NSString *newRecognizedString;
1009 
1010     newRecognizedString = @"";
1011 
1012     for (int i = 0; i<[singlePercentArray count]; i++) {
1013 
1014         if ([[singlePercentArray objectAtIndex:i] doubleValue] > 60.0) {
1015 
1016             newRecognizedString = [NSString stringWithFormat:@"%@%@",newRecognizedString,
1017                                    [[confidence objectAtIndex:i] text]];
1018 
1019         }
1020 
1021     }
1022 
1023     return newRecognizedString;
1024 
1025 }
```

Abbildung 33

Bei Aufruf der Methode wird ein Array übergeben („singlePercentArray“), welches die Prozentwerte der OCR-Genauigkeit beinhaltet. Außerdem wird ein weiteres Array an die Methode übergeben („oldRecognizedStringArray“), welches die ermittelte Seriennummer als einzelne Zeichen enthält. Diese Nummer ist natürlich nicht korrekt, da sie mindestens ein Zeichen mehr enthält, als die Vorgabe im Testdeck.

Zu Beginn wird ein neuer String „newRecognizedString“ angelegt. In einer „for“-Schleife wird jeder Prozentwert im „singlePercentArray“ überprüft. Ist der Prozentwert größer 60%, ist das Zeichen mit hoher Wahrscheinlichkeit korrekt und somit wird das dazugehörige Zeichen dem String „newRecognizedString“ hinzugefügt. Sollte der Prozentwert kleiner der 60% sein, so wird das dazugehörige Zeichen nicht in den String geschrieben.

Nach Durchlauf aller Zeichen und derer Prozentwerte, wird der String mit der gekürzten Seriennummer zurückgeliefert.

5 Fazit

Rückblickend auf das Projekt FitnessVoter wurde eine Vielzahl von Themenbereichen abgedeckt. Zu Beginn musste eine Vorrichtung entwickelt werden, welche den Prozess des Bewertens von Banknoten erleichtert.

Neben dem Programmieren in der Objective-C Umgebung wurde auch der Bereich der Bildverarbeitung benötigt. Das optimale Aufbereiten der Bilder für die Texterkennung ist essenziell für gute OCR-Ergebnisse. Auch die grafische Gestaltung der Benutzeroberfläche ist wichtig bei einem solchem Projekt um den Smart Faktor für den User zu gewährleisten.

Um die Daten auch extern sichern zu können, musste ein Webserver eingerichtet werden, welcher mittels PHP die Daten in einer MySQL-Datenbank speichert. Dazu ist es ebenfalls notwendig eine Verbindung zwischen den beiden Geräten herzustellen, wozu der Bereich der Netzwerktechnik benötigt wurde.

5.1 Erkenntnisse zum Programmablauf und zur Voting-Vorrichtung

Der Prototyp der entwickelten Voting-Vorrichtung ist stabil und erfüllt seine Aufgaben ohne Probleme. Allerdings ist ein Transport nur im demontierten Zustand gut zu gewährleisten. Außerdem ist das Gewicht der einzelnen Komponenten für einen längeren Transport ohne Koffer oder Ähnlichem zu hoch. Jedoch lässt sich das Apple iPad in der Vorrichtung sowohl im Stehen, als auch im Sitzen gut bedienen.

Der Programmablauf wurde selbst ausgearbeitet und enthält alle Anforderungen, die für das Projekt im Vorfeld definiert wurden. Er wurde eins zu eins in der Applikation umgesetzt und läuft stabil. Allerdings könnte der Programmablauf durch kleine Änderungen im Bereich der Dauer eines Voting-Durchlaufes verbessert werden.

5.2 Erkenntnisse zur Bedienung

Vor dem Benutzen des Programms, sollte das iPad stets in der Vorrichtung korrekt platziert werden. Dabei sollte auch die angebrachte Beleuchtung eingeschaltet werden.

Die Bedienung der App als User ist selbsterklärend, da immer nur die Funktionen durchführbar sind, welche augenblicklich benötigt werden. Mittels der Integration vieler Icons ist ein grafischer Leitfaden implementiert worden, welcher dem User hilft im Programm voranzuschreiten (Smart-Faktor). Die Bewertung einer Banknote kann schon mit 2 betätigten Buttons abgeschlossen werden.

Um die Applikation als Administrator nutzen zu können, benötigt man einen PIN um in das Administrationsmenü zu gelangen. Zum Datenaustausch zwischen Apple iPad und PC sind ebenfalls Vorbereitungen nötig. Neben dem Host-Namen bzw. der IP-Adresse des Webservers wird auch eine bestehende Verbindung der beiden Geräte benötigt. Danach besteht allerdings die Möglichkeit, das Tablet komplett für den User zu konfigurieren.

5.3 Schlussfolgerung der Erkenntnisse

Der Prototyp der Voting-Vorrichtung ist funktionell aber kann im Bereich des Transports noch verbessert werden. Eine Verbesserung ist ebenso beim Programmablauf der Applikation denkbar. Dabei sollte die Dauer beim Auslesen und Prüfen der Seriennummer verkürzt werden.

Die Bedienung der Software als Benutzer ist übersichtlich und benötigt keine größeren Vorbereitungen, wodurch der User die Konzentration auf den Zustand der Banknote gerichtet lassen kann. Außerdem muss sich der Benutzer nicht durch mehrere Seiten durcharbeiten, sondern hat die Bewertung auf der selben View, wie auch die Anzeige der Banknote auf dem Bildschirm.

Der Administrator muss bestimmte Eingaben tätigen, hat aber danach die Möglichkeit die App für den User komplett einzurichten.

6 Ausblick

Das bisher entwickelte System zur Bewertung von Banknoten wird als Prototyp verstanden. Die implementierten Funktionen sind die Grundbausteine zur Bewertung und Erfassung der Banknoten. Hierbei besteht jedoch die Möglichkeit der Erweiterung der Funktionalität der App. Aber nicht nur die Software kann erweitert werden, auch die Voting-Vorrichtung ist noch im Musterzustand und bietet kleinere Verbesserungen.

6.1 Ausbau der Applikation

Die App bietet noch verschiedene Möglichkeiten der Erweiterung an. Beispielsweise wäre eine automatische Erkennung der Banknote auf der Geldablage der Vorrichtung denkbar, welche ohne Betätigen eines Buttons ein Bild erstellt. Dies kann mittels der Kantendetektion realisiert werden. Desweiteren wäre das Aufnehmen beider Seiten der Banknote als Bild eine weitere Möglichkeit zur Erweiterung, da auf der Rückseite Beschädigungen oder Verschmutzungen vorhanden sein können.

Auch die Bewertung bietet noch Ausbaugelegenheiten. Dabei wäre zum Beispiel ein schnellerer Ablauf durch die Optimierung der Texterkennung vorstellbar. Hierzu müssten die Bereiche, in denen die Seriennummer enthalten ist, besser für die OCR vorbereitet werden. Auch das Einzeichnen der Verschmutzungen oder Beschädigungen auf dem erstellten Bild des Geldscheines ist denkbar. Dies würde auf der Hauptseite der Bewertung geschehen und mittels Touch-Funktion realisiert werden.

Eine weitere Abwandlung der Applikation wäre das Auslagern der Erstellung der Banknotenbilder mit dem iPad. Dabei würden die Bilder von dem Webserver geladen und dem User der Reihe nach angezeigt. Dabei würde der User lediglich die Geldscheine bewerten und müsste sich weder um die Erstellung der Bilder noch um die Korrektheit der Seriennummer kümmern. Danach müssten nur die Bewertungen an den Webserver übermittelt werden, wo sie den bereits bestehenden Testdecks mit erfassten Seriennummern und Bildern zugeordnet werden.

6.2 Überarbeitung der Voting-Vorrichtung

Der Prototyp der Voting-Vorrichtung erfüllt seinen Zweck, hat aber noch Raum für Verbesserungen. Dieser besteht aus einer großen Grundplatte aus Holz, welche Verkleinert werden könnte um Gewicht einzusparen. Außerdem könnte eine andere, besser aussehende Holzart verwendet werden, damit die Vorrichtung nicht nur den Nutzen, sondern auch der Optik entspricht. Da diese Vorrichtung mit den LED-Streifen ausgestattet ist und öfter transportiert werden soll, wäre das Anfertigen einer Transportbox eine gute Lösung Beschädigungen zu vermeiden.

Um einen optimalen Programmablauf zu gewährleisten, wurde die Geldablage des Prototyps mit schwarzen Buntpapier verkleidet. Dies könnte eleganter gelöst werden, indem die Geldablage mit einem schwarzen Lack überzogen wird. Außerdem könnte ein Lichtschutz um die Geldablage angebracht werden, um den direkten Einfall von Sonnenlicht oder anderen Lichtquellen aus der Umgebung zu verhindern.

Zur technischen Verbesserung der Vorrichtung wäre ein Kabelkanal für den Anschluss der Beleuchtung denkbar. Somit wären gleichzeitig die Kabel bei einem Transport geschützt. Des Weiteren ist eine, in der Vorrichtung integrierte und an das Stromnetz des Prototypen angeschlossene, Steckdose vorstellbar. Über diese hätte der User die Möglichkeit das Apple iPad auch während der Nutzung zu Laden.

Literatur

[V01] Dipl.-Ing. (FH), M. Sc. Thomanek, Rico:

Vorlesungsunterlagen: Grundlagen der iOS-Programmierung, Hochschule Mittweida, 2014

[V02] Prof. Dr.-Ing. habil. Lutz Winkler, Dipl.-Ing. (FH), M. Sc. Thomanek, Rico:

Vorlesungsunterlagen: HTML5 Hyper Text Markup Language 5, Hochschule Mittweida, 2015

[B01] Lars Schulten, O`Reillys Taschenbibliothek, Objective-C – kurz & gut, O`Reilly Verlag, 1. Auflage, April 2013

[B02] Max Seelemann, Objective-C kompakt, Ein Kurs für Umsteiger und Fortgeschrittene, dpunkt.verlag, Dezember 2011

[B03] Giesbert Damaschke, PHP & MySQL, Der Web-Baukasten für Einsteiger und Individualisten, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, 1. Auflage 2014

[Web01] GitHub – GPUImage von Brad Larson

https://github.com/BradLarson/GPUImage

[Web02] Lehrunterlagen PHP – Prof. Dr.-Ing. habil. Lutz Winkler

http://www.global.hs-mittweida.de/~telecom/html3/

[Web03] Lehrunterlagen CSS – Prof. Dr.-Ing. habil. Lutz Winkler

http://www.global.hs-mittweida.de/~telecom/html3/css/index.htm

[Web04] Stack overflow – Forum

http://stackoverflow.com/

Anlagen

Benutzerhandbuch der App - User

Dieses Benutzerhandbuch soll die Bedienung der App für den User noch leichter gestalten. Dafür sind im Folgenden alle Funktionen, welche die Applikation für den Benutzer bereitstellt, zusammengefasst und kurz erläutert.

Zu Beginn der Benutzung, sollte sichergestellt werden, dass das iPad korrekt in der Votingvorrichtung platziert wurde und die LED-Streifen an der Vorrichtung in Betrieb sind. Anschließend kann die App, durch Auswählen des FitnessVoter Icons gestartet werden.

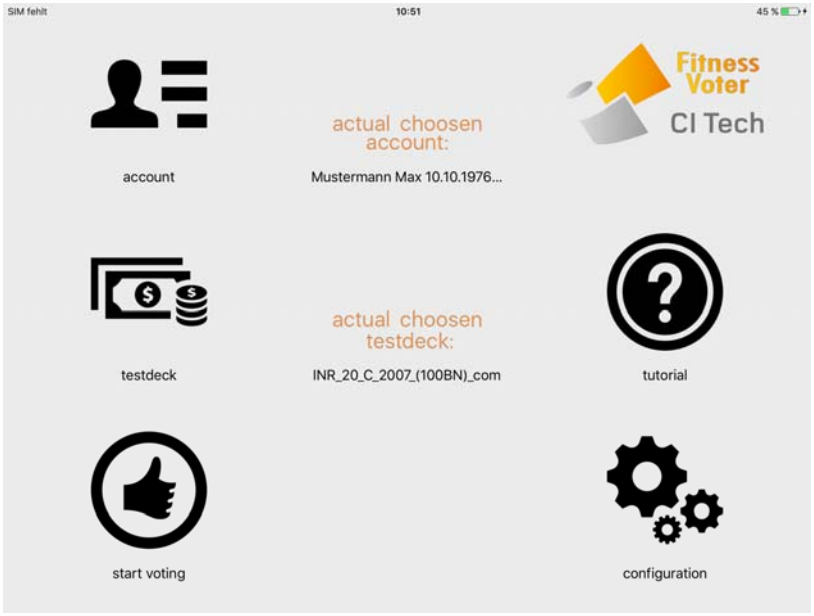
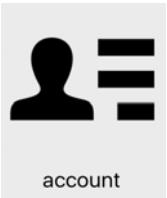


Abbildung 34

Auf der Startseite der Software, hat der User mehrere Auswahlmöglichkeiten. Neben „account“, „testtdeck“, „start voting“ und „tutorial“ ist auch der Punkt „configuration“ aufgeführt. Allerdings ist dieser lediglich für die Konfiguration der Applikation gedacht und somit nur für den Administrator zugänglich, da dieses Menü durch einen PIN geschützt ist.



Im Menüpunkt „account“ muss der User einen Account auswählen, welcher der Bewertung im späteren hinzugefügt wird. Dabei hat er die Möglichkeit ein bereits bestehendes Profil auszuwählen, zu Bearbeiten oder zu Löschen. Es besteht allerdings auch die Möglichkeit einen neuen Account anzulegen.

Wird der Menüpunkt „testdeck“ gewählt, so steht die Wahl eines bereits erstellten Testdecks an. Ein Testdeck wird immer, zum jeweils zu bewertenden Bündel Banknoten vom Administrator erstellt. Auch diese Auswahl muss getroffen werden, bevor das Bewerten der Geldscheine möglich ist.

Durch Auswahl von „start voting“ wird das Bewerten der Geldscheine, nach Auswahl eines Accounts und eines Testdecks, gestartet.

Der Menüpunkt „tutorial“ öffnet eine neue View, welche zur Erläuterung des Votingvorgangs dient. Dabei wird in einzelnen Schritten erklärt, was beim Bewerten der Banknoten zu beachten ist.

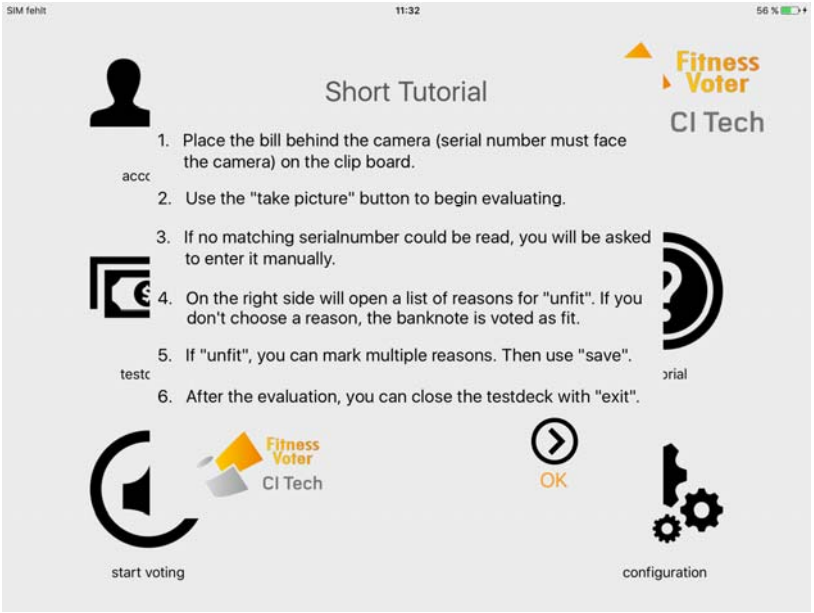


Abbildung 35

Bevor das Bewerten der Banknoten durchgeführt werden kann, müssen nun zuerst ein Account und ein Testdeck ausgewählt werden. Die aktuell ausgewählten Werte für Account und Testdeck, werden auf der Startseite angezeigt. Zum Ändern muss der „account“-Button betätigt werden, damit sich eine neue View öffnet. Diese besteht aus einer Tabelle und fünf Funktionsbuttons.

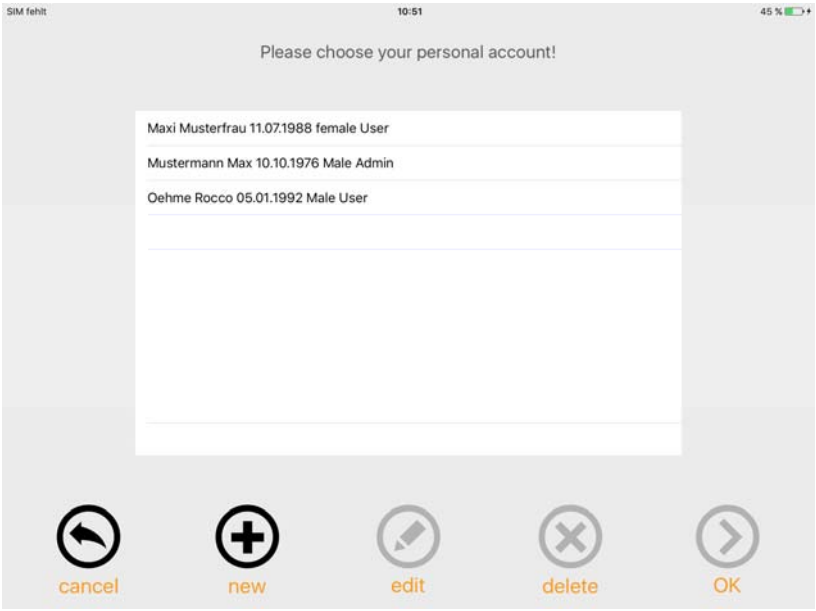
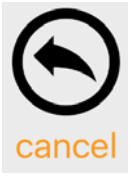


Abbildung 36

In der Tabelle werden alle bereits bestehenden User-Profile aufgelistet. Diese können per Berührung ausgewählt werden und mittels der Funktionstasten verarbeitet werden. Ist kein Account ausgewählt, so sind die Buttons „edit“, „delete“ und „next“ deaktiviert.

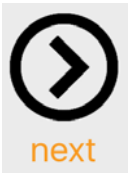


Mittels „cancel“ besteht die Möglichkeit die „account“-View zu verlassen. Dabei gelangt man zur Startseite zurück.

Durch Betätigen des „new“ Buttons öffnet sich ein kleines neues Fenster, welches wie ein Formular aufgebaut ist um einen neuen Account anzulegen. Nach dem Ausfüllen der Textfelder wird mittels des „save“-Buttons ein neuer Account erzeugt. Dieser erscheint anschließend in der Tabelle.

Der „edit“ Button ist nur auf einen bereits bestehenden Account anwendbar. Dabei wird ein ähnliches Fenster geöffnet wie bei „new“. Jedoch sind in den Textfeldern die Informationen des zu ändernden Profils enthalten und können ersetzt werden. Mittels „save“ werden die Angaben in der Tabelle aktualisiert.

Wie auch „edit“ ist „delete“ nur anwendbar, wenn vorher ein Account ausgewählt wurde. Durch Betätigen des „delete“ Buttons wird das aktuell ausgewählte Profil gelöscht und die Tabelle wird aktualisiert.



Der „next“ Button bewirkt das Speichern des ausgewählten Accounts für das anstehende Bewerten der Banknoten. Dabei wird automatisch die View geschlossen und zur Startseite gewechselt.

Nachdem der User sein Profil ausgewählt hat, fehlt noch die Auswahl eines Testdecks. Mit dem „testdeck“ Button auf der Startseite wird eine neue View geöffnet.

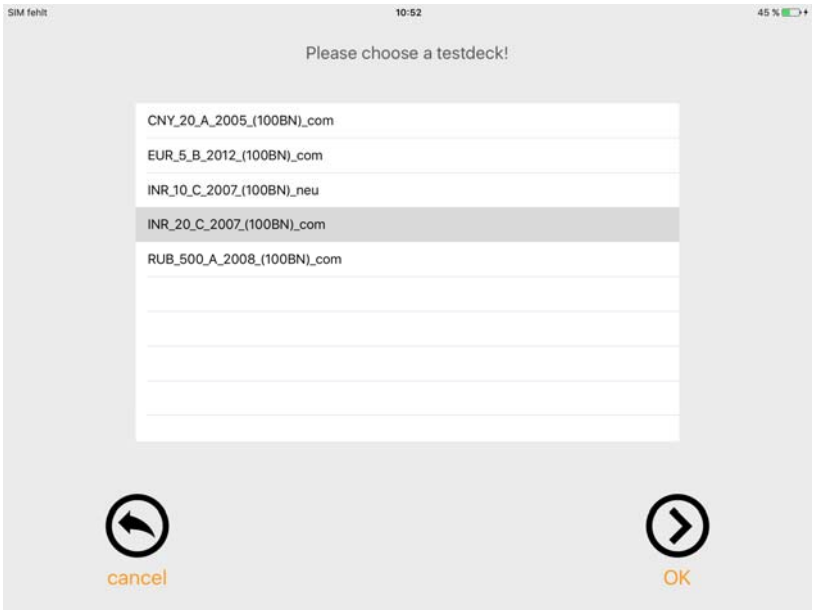


Abbildung 37

Diese View besteht ebenfalls aus einer Tabelle und Funktionsbuttons. In der Tabelle werden alle, vom Administrator angelegten, Testdecks angezeigt. Diese Testdecks enthalten essenzielle Informationen für das Voting und müssen somit vorher ausgewählt werden.

Mittels „cancel“ besteht die Möglichkeit die „testdeck“-View zu verlassen. Dabei gelangt man zur Startseite zurück.

Der „next“ Button ist nur aktiviert, wenn vorher ein Testdeck aus der Liste ausgewählt wurde. Durch das Betätigen dieses Buttons wird das Testdeck für das Voting gespeichert und alle wichtigen Informationen aus dem Testdeck weitergeleitet. Außerdem gelangt man zurück auf die Startseite.

Nachdem ein Account und ein Testdeck ausgewählt wurden, kann das Voting starten. Durch Auswahl des „start voting“ Menüpunktes auf der Startseite gelangt man nun zu der Hauptseite des Voting-Prozesses.

Die Hauptfunktion der Applikation besteht im Bewerten von Banknoten. Um diese Banknoten zu erfassen soll neben einem Bild des Geldscheins auch die Seriennummer ausgelesen werden. Der Ausgangspunkt des Bewertens ist die folgende View.

Abbildung 38

Diese besteht aus einer Live-View der Banknote, einem Zähler der bewerteten Banknoten, sowie zwei Funktionstasten.

Mittels des „exit“ Buttons kann das Voting beendet werden, wobei gleichzeitig das Testdeck, sobald eine Bewertung stattgefunden hat, gespeichert wird. Die Live-View der Banknote soll helfen, den Geldschein gut zu platzieren. Wurde die Banknote platziert drückt der User den Button „take picture“ um den Vorgang zu starten. Danach wird ein Bild erzeugt und die Texterkennung wird gestartet.

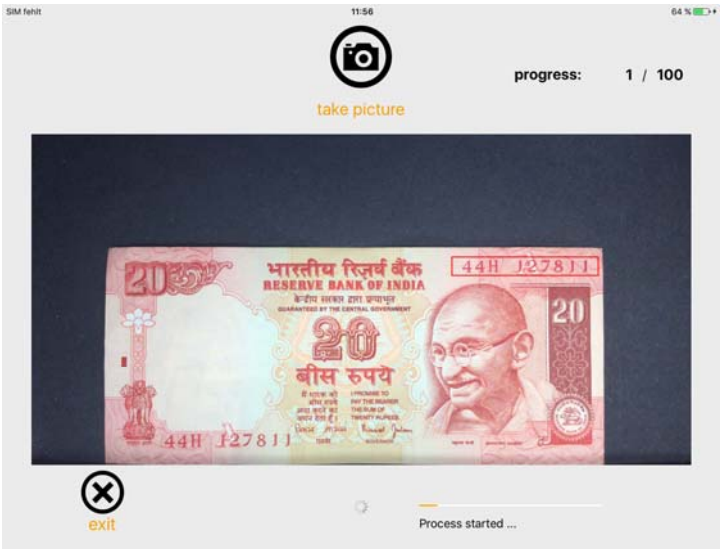


Abbildung 39

Im unteren Bereich der View erscheint ein Balken, welcher den Fortschritt der Texterkennung anzeigt. Außerdem wird dem User mit roten Rechtecken verdeutlicht, an welchen Stellen eine Seriennummer ausgelesen wird. Bei 2 Seriennummern auf einer Banknote werden diese immer abwechselnd ausgelesen.

Abbildung 40

Nach Fertigstellung der Texterkennung erscheint im oberen Bereich der View eine neue Anzeige und der Fortschrittsbalken der OCR ist vollständig ausgefüllt. Dabei werden die ausgelesenen Ergebnisse und zwei neue Button angezeigt. Des Weiteren werden die Bereiche der Seriennummern verschieden farbig auf dem Bild gekennzeichnet. Ist die ausgelesene Seriennummer allerdings nicht korrekt, so können mittels „Edit“ Änderungen vorgenommen werden.

Abbildung 41

Durch betätigen des „Edit“ Buttons erscheint auf der rechten Seite der neuen Ansicht eine Tabelle in der Vorschläge, beziehungsweise alle ermittelten Nummern, welche der Syntax entsprechen, vorhanden sind. Die Seriennummer kann aber auch händig in das Textfeld in der Mitte der neuen Ansicht eingegeben werden.

Auf der rechten Seite des Bildschirms befindet sich eine Liste mit verschiedenen Gründen für einen schlechten Zustand der Banknote. Diese einzelnen Punkte sind auswählbar. Ist die Banknote allerdings in einem guten Zustand, darf kein Grund ausgewählt werden. Somit wird im oberen Balken auch die Ausschrift „Fit“ angezeigt.

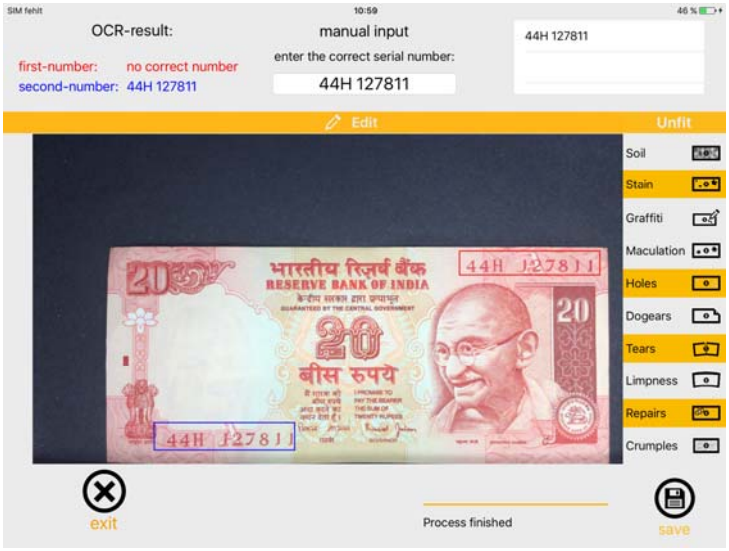


Abbildung 42

Sollte sich die Banknote allerdings in einem unfitten Zustand befinden, so muss in der Liste eine Auswahl getroffen werden. Dabei sind die Gründe für den schlechten Zustand des Geldscheines zu nennen, wobei auch eine Mehrfachnennung möglich ist. Sobald eine Auswahl getroffen wurde, ändert sich der Text im oberen Balken von „Fit“ zu „Unfit“.

Durch Betätigen des „save“ Buttons werden alle Daten zu dem gerade bewerteten Geldschein zwischengespeichert und man gelangt zur Hauptseite des Votings zurück.

Abbildung 43

Die Anzeige der bewerteten Banknoten zählt weiter und die Live-View der Banknote ist wieder zusehen. Mittels „take picture“ kann mit dem Testdeck fortgefahren werden oder durch Drücken des „exit“ Buttons kann das Testdeck beendet werden, wobei alle Daten gespeichert werden. Beendet der User allerdings das Bewerten, so gelangt er zurück zum Hauptmenü.

Dabei werden im Hauptmenü die Daten vom letzten Voting angezeigt. Der Account und das Testdeck, mit welchen die letzte Bewertung durchgeführt wurde werden als „actual choosen“ ausgegeben. Diese müssen bei Wiederholung der Bewertung mit gleichen Parametern nicht erneut ausgewählt werden.

Benutzerhandbuch der App - Administrator

Da, vor und nach der Benutzung der Applikation im User-Modus, einige Einstellungen getroffen werden müssen, wurde ein PIN-geschützter Bereich in der App eingerichtet. Dieser Bereich dient zur Konfiguration und Verwaltung der Testdecks.

Um die Funktionen, die dem Administrator vorbehalten sind, auszuführen, muss auf der Startseite der Menüpunkt „configuration“ ausgewählt werden.

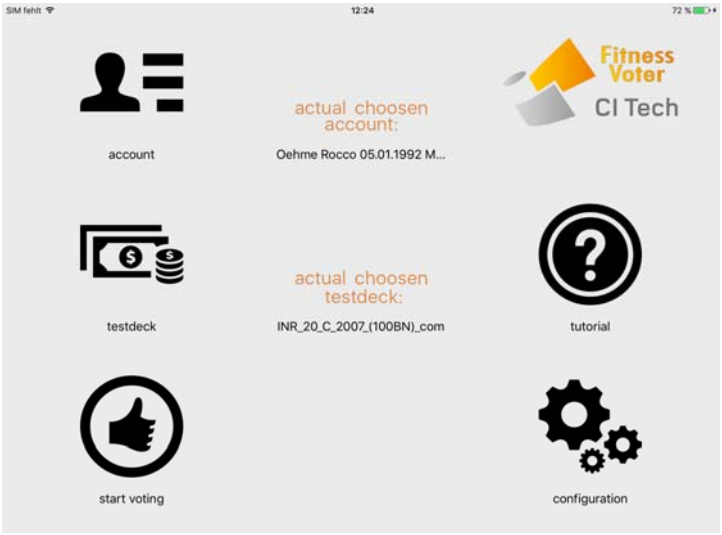


Abbildung 44

Danach wird der Administrator aufgefordert den PIN einzugeben. Dieser ist Standardmäßig auf „0000“ gesetzt.

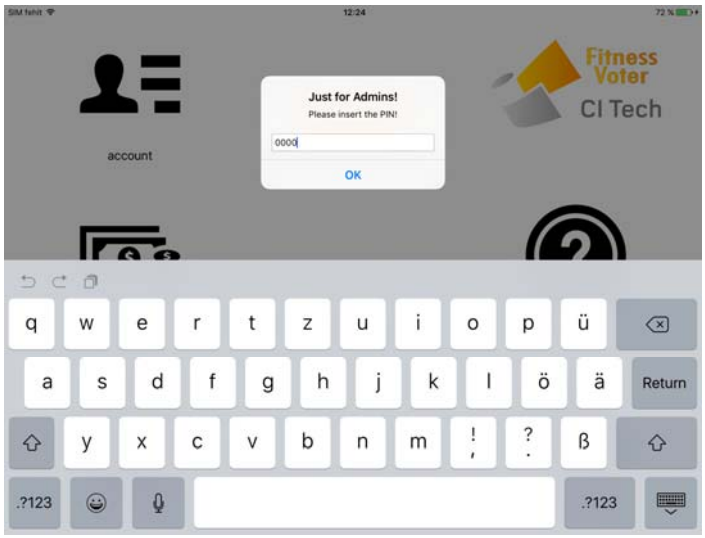


Abbildung 45

Nach der Eingabe und dem Bestätigen durch den „OK“ Button wird das Administrationsmenü geöffnet.

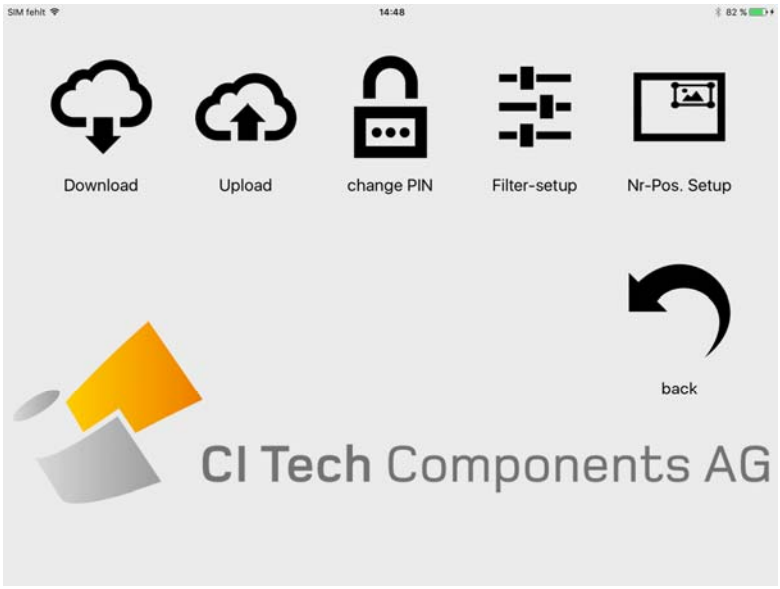
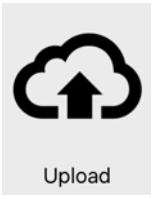


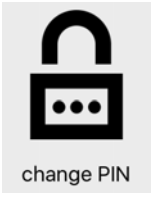
Abbildung 46

In diesem Menü hat der Administrator eine große Auswahl an Funktionen.

Hinter dem Button „Download“ befindet sich eine Auswahl der bereits auf dem Server erzeugten Testdecks. Diese können auf das Apple iPad heruntergeladen und somit dem User zur Bewertung zur Verfügung gestellt werden.

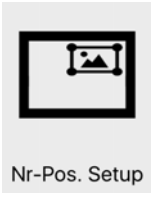


Mittels „Upload“ besteht die Möglichkeit, alle fertig bewerteten Testdecks an den Server zu senden, wo diese dann verarbeitet werden.



Der Menüpunkt „change PIN“ ist zum Ändern des voreingestellten Schlüssels zur Anmeldung für das Administrationsmenü vorgesehen.

In „Filter-setup“ hat der Administrator die Möglichkeit mittels einer Live-View auf die Banknote verschiedene Filtereinstellungen zu testen. Dabei werden die Parameter für die Filtereinstellungen ausgegeben, welche bei der Erstellung eines Testdecks notwendig sind.



Der „Nr-Pos. Setup“ Button führt ebenfalls zu einer Seite mit Live-View. Hier können die Schwellwerte für die Kantendetektion und die Größe und Koordinaten der Seriennummer getestet werden.

Mittels „back“ gelangt der Administrator wieder zurück zur Startseite der Applikation.

Nach Aufruf des Download Buttons, öffnet sich eine View, die aus einem Header-Bereich, einer Tabelle und zwei Funktionsfeldern besteht.

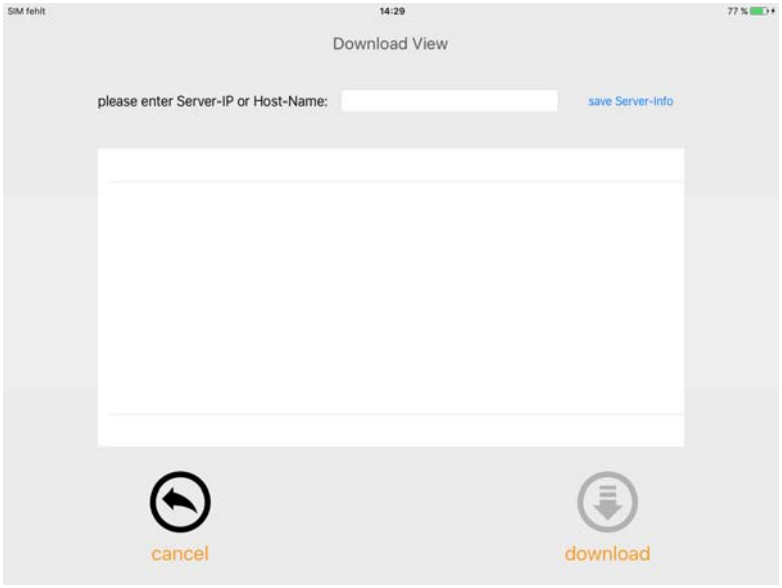


Abbildung 47

Im Header-Bereich muss entweder die IP-Adresse oder der Host-Name des Webservers angegeben werden, auf dem die erstellten Testdecks vorhanden sind.

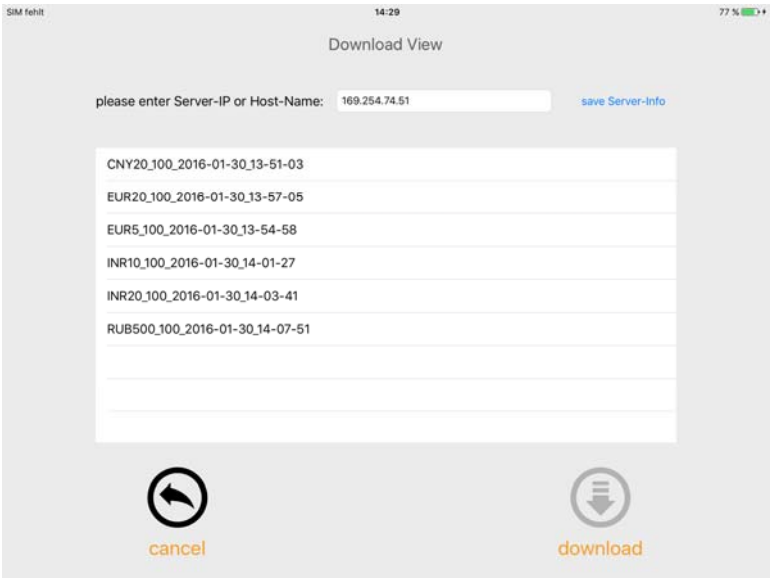


Abbildung 48

Danach erscheinen alle Testdecks, welche bis zu diesem Zeitpunkt auf dem Server erzeugt wurden, in der Tabelle. Nach Auswahl eines Testdecks wird der „download“ Button

aktiviert. Betätigt der Administrator diesen, so wird das ausgewählte Testdeck vom Server heruntergeladen. Danach färbt sich die Zeile der Tabelle grün.

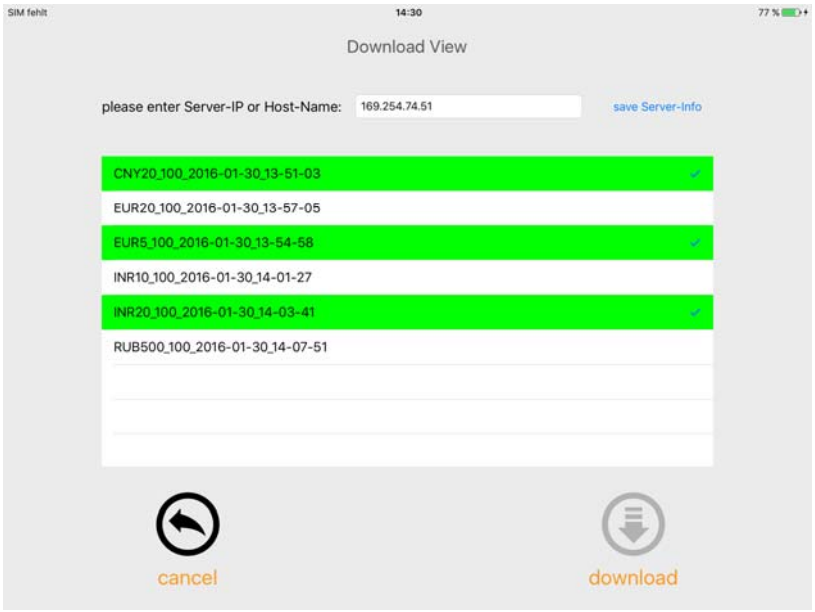


Abbildung 49

Nun ist das Testdeck in der Applikation gespeichert und wird für den User in der Wahl des Testdecks zur Bewertung der Banknoten sichtbar. Der Button „cancel“ führt wieder zurück zum Administrationsmenü.

Der Menüpunkt „Upload“ bietet eine weitere Möglichkeit die Testdecks zu verwalten. Dazu öffnet sich eine neue View, welche ähnlich der „Download“ View aufgebaut ist.

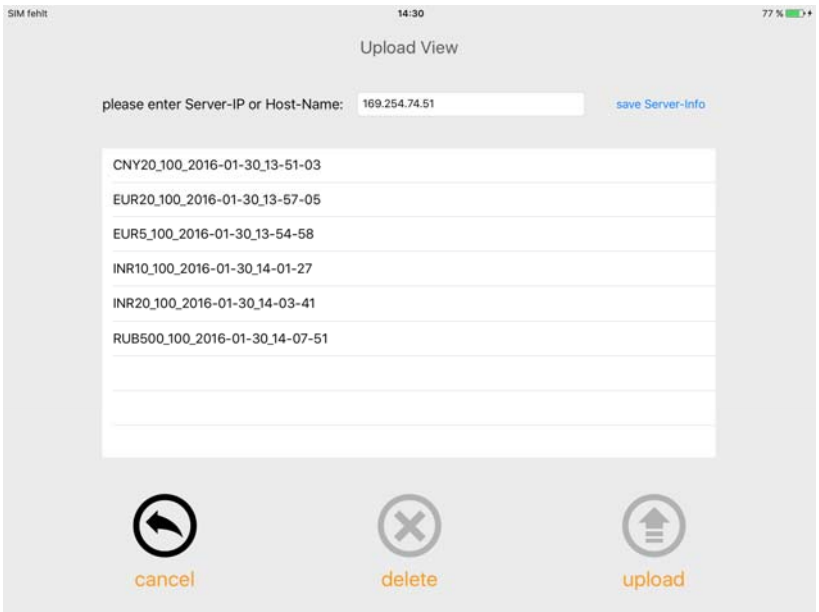


Abbildung 50

Allerdings ist auf dieser Seite ein weiterer Button „delete“ hinzugefügt. Dieser ist, wie auch der „upload“ Button zu Beginn deaktiviert. Diese beiden Buttons werden erst aktiv, wenn ein Testdeck aus der oberen Liste ausgewählt wurde. In dieser Liste sind alle bereits abgeschlossenen Testdecks enthalten. Mittels „upload“ wird das gewählte Testdeck an den Server übergeben und die Zeile wird grün gefärbt.

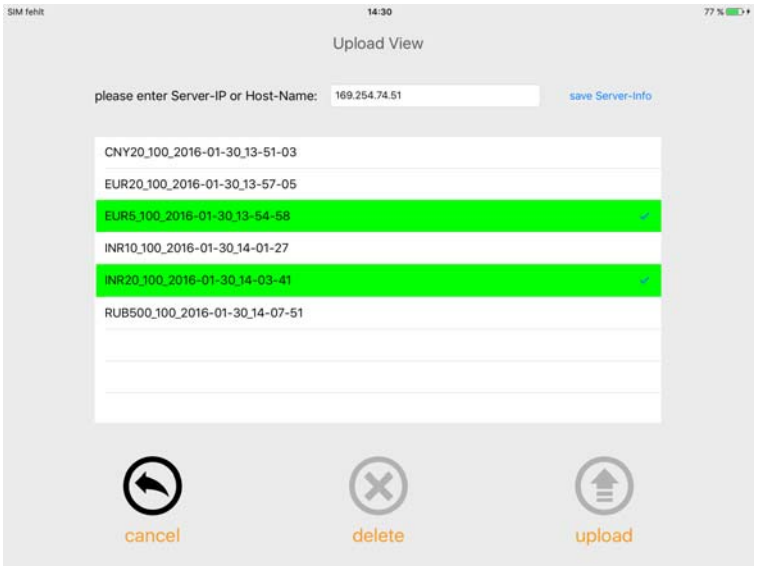


Abbildung 51

Natürlich besteht auch die Möglichkeit die bereits hochgeladenen Testdecks auf dem Apple iPad mit dem „delete“ Button zu löschen. Wählt der Administrator „cancel“ gelangt er wieder in das Administrationsmenü.

Zum Ändern des Schlüssels für die Authentifizierung des Administrationsmenüs ist der Menüpunkt „change PIN“ vorgesehen. Dabei erscheint ein kleines Fenster mit einer Art Formular.

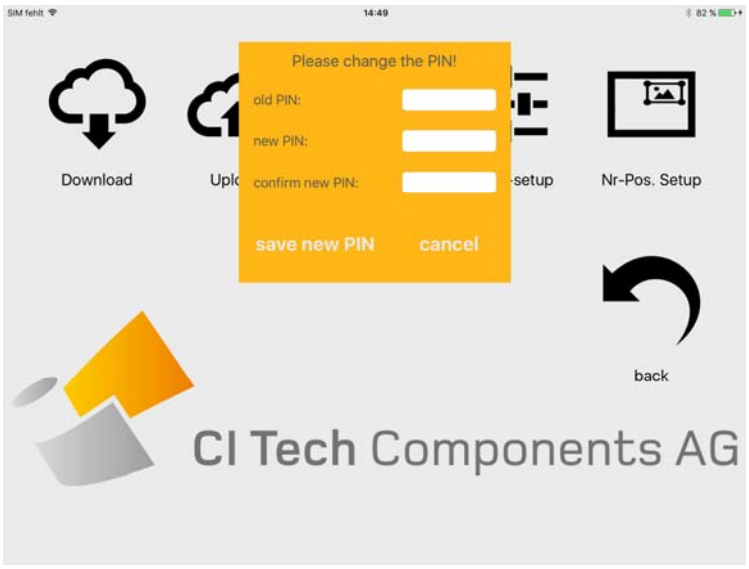


Abbildung 52

Nach der Eingabe des aktuellen PIN´s muss noch zweimal der neue gewünschte PIN eingegeben werden. Ist der alte Schlüssel korrekt und stimmen die beiden Neuen überein, so wird der PIN geändert.

Als Hilfestellung zur Erstellung eines neuen Testdecks auf dem Webserver, wurden die folgenden zwei Menüpunkte zum Administrationsmenü hinzugefügt. Mittels „Filter-Setup“ hat der Administrator die Möglichkeit verschiedene Filtereinstellung auf einer Live-View der Banknote anzuwenden.

Abbildung 53

Dabei sind zwei unterschiedliche Filter nutzbar. Zum einen ist ein Farbfilter, und zum anderen ist ein Luminancefilter anwendbar.

Zum Anwenden der Filter auf die Live-View der Banknote muss der jeweilige ON/OFF-Switch aktiviert werden.

Abbildung 54

Ist der Farbfilter aktiviert, kann der Administrator zwischen den Farben „red“, „green“ und „blue“ auf der linken Seite der View wählen. Dabei wird ein Graubild erzeugt. Aber der Administrator kann ebenfalls ein Farbbild erhalten, indem er auf die Option „all colors“ wechselt. Mittels der Slider können die RGB-Werte verändert werden. Die Änderungen werden direkt auf die Live-View angewendet.

Nachdem alle Einstellungen für den Farbfilter getroffen wurden, kann auch der Luminancefilter eingeschaltet werden.

Abbildung 55

Somit entsteht ein Schwarz-Weiß Bild. Dieses kann durch Verschieben des Sliders so angepasst werden, dass die Seriennummer auf dem Bild ohne Hintergrundeinwirkungen gut erkennbar ist. Der Slider stellt dabei den Schwellwert des Filters dar. Mittels der beiden Buttons in der unteren rechten Ecke kann eine Feinabstimmung des Schwellwertes getroffen werden.

Sind die Einstellungen so getroffen, dass der Administrator sie für optimal befindet, können die Werte abgelesen und in das Formular zur Erstellung eines neuen Testdecks übertragen werden.

Im Menüpunkt „Nr-Pos. Setup“ ist ebenfalls eine Live-View der Banknote zu sehen. Hierbei werden Werte für die Kantendetektion und Position und Größe der Seriennummer getestet.

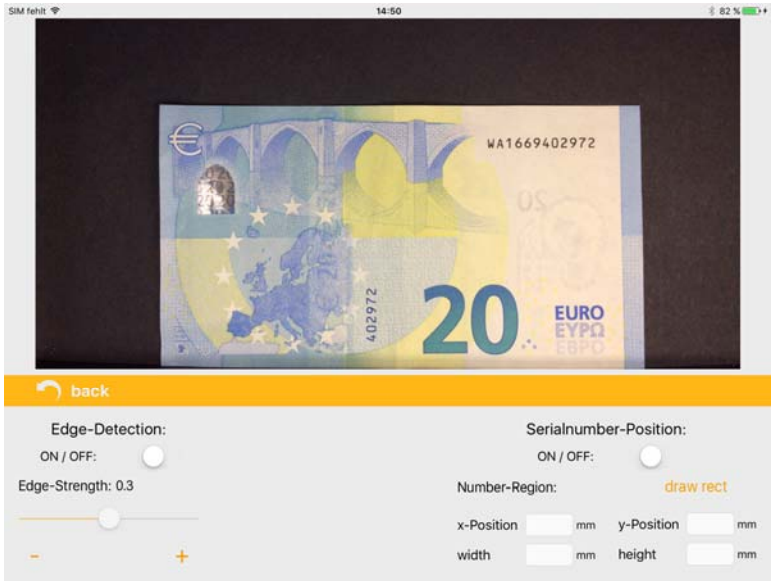


Abbildung 56

Nach dem Öffnen dieser Seite sind beide Optionen deaktiviert. Zum Aktivieren der Kantendetektion muss der ON/OFF-Slider betätigt werden. Danach wird der Filter direkt auf die Live-View angewendet.

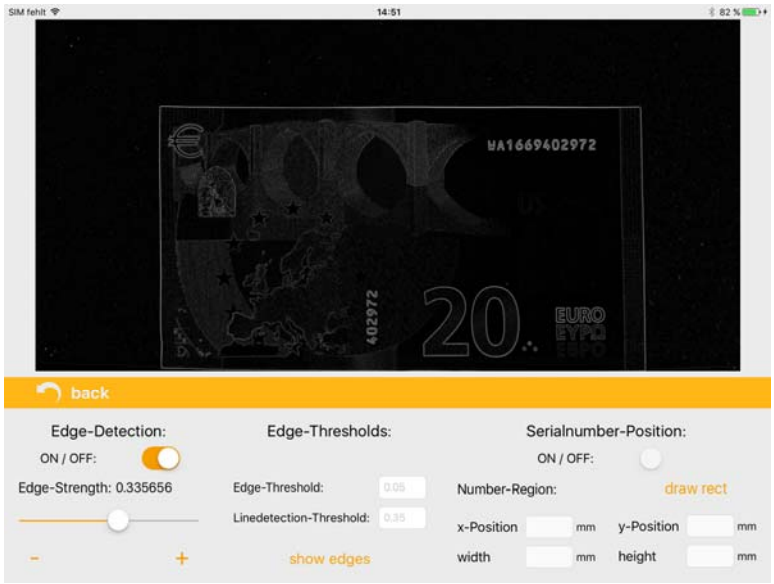


Abbildung 57

Der „SobelEdgeDetectionFilter“ wandelt alle Pixel des Bildes, die unter dem Edge-Strength Wert liegen in schwarze Pixel. Alle darüberliegenden Pixel werden weiß. Auch hier besteht für den Administrator die Möglichkeit mittels des Plus- und Minus-Buttons eine Feinabstimmung vorzunehmen. Danach müssen noch zwei Werte für die Kantenschwelle und Linienermittlung angegeben werden. Diese sind standardmäßig 0.05 für die Kantenschwelle und 0.35 für die Linienermittlung. Danach können die ermittelten Kanten durch Betätigen des „show edges“ Button angezeigt werden.

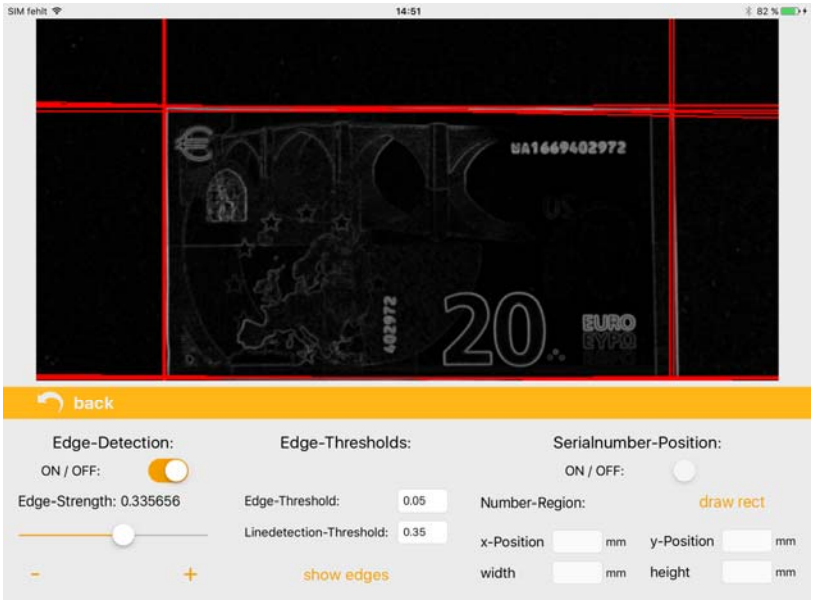


Abbildung 58

Das Ergebnis sollte es sein, dass an jeder Kante des Geldscheines mindestens eine rote Linie gezeichnet wurde. Dies bedeutet, dass die Kante ermittelt wurde. Die eingestellten Werte müssen danach wieder in das Testdeck eingetragen werden.

Anhand der ermittelten Kanten, kann die abgemessene Position und Größe der Seriennummer auf der Banknote überprüft werden. Dazu muss die Kantendetektion deaktiviert werden und die Seriennummer-Position aktiviert werden.

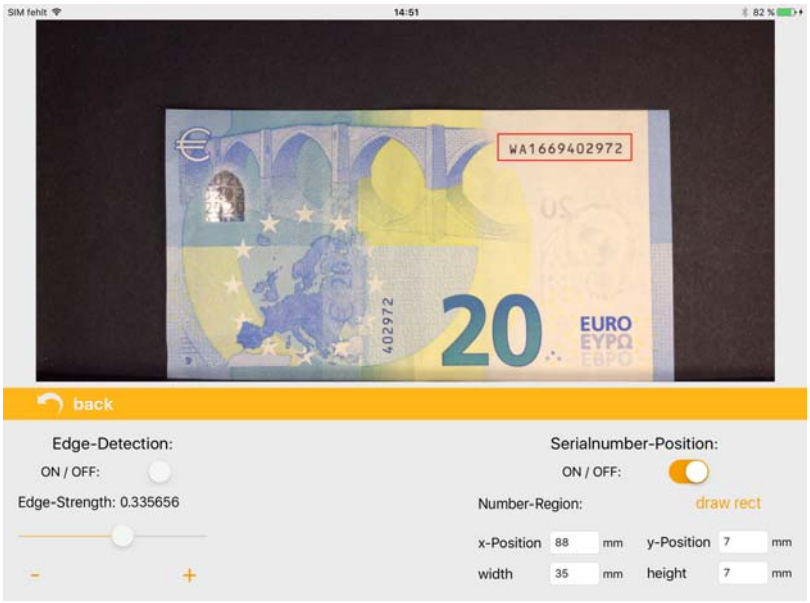


Abbildung 59

Danach können die vier Werte für X- und Y-Koordinate, Breite und Höhe eingegeben werden. Diese Werte sind alle als Millimeter-Wert anzugeben. Durch Betätigen des „draw rect“ Buttons wird ein rotes Rechteck über die Banknote gelegt. Somit können die gemes-

senen Werte überprüft werden. Mittels „back“ gelangt der Administrator wieder zum Administrationsmenü zurück.

Verlässt der Administrator auch das Administrationsmenü mit „back“, so befindet sich die App wieder auf der normalen Startseite und kann vom User zur Bewertung von Geldscheinen verwendet werden.

Verbindungseinrichtung PC - Apple iPad

Um eine unkomplizierte Verbindung zwischen den beiden Geräten einzurichten, bietet es sich an ein Ad-hoc Netzwerk mittels der Drahtlosen Netzwerkkarte des Windowsbasierten Notebooks zu erstellen. In diesem Netzwerk meldet sich das Apple iPad an und kann somit über das Netz mit dem Laptop kommunizieren und Daten austauschen.

Das Erstellen des Ad-hoc Netzwerkes ist in wenigen Schritten umsetzbar. Zuerst öffnet man über die Windows Start-Schaltfläche die „Systemsteuerung“.

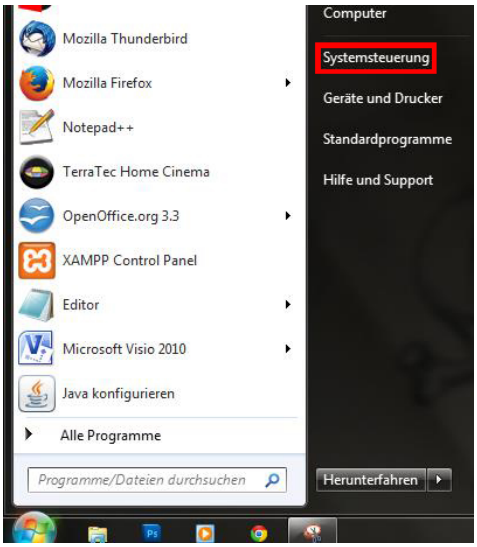


Abbildung 60

Danach wählt man den Menüpunkt „Netzwerk und Internet“ aus um fortzufahren.

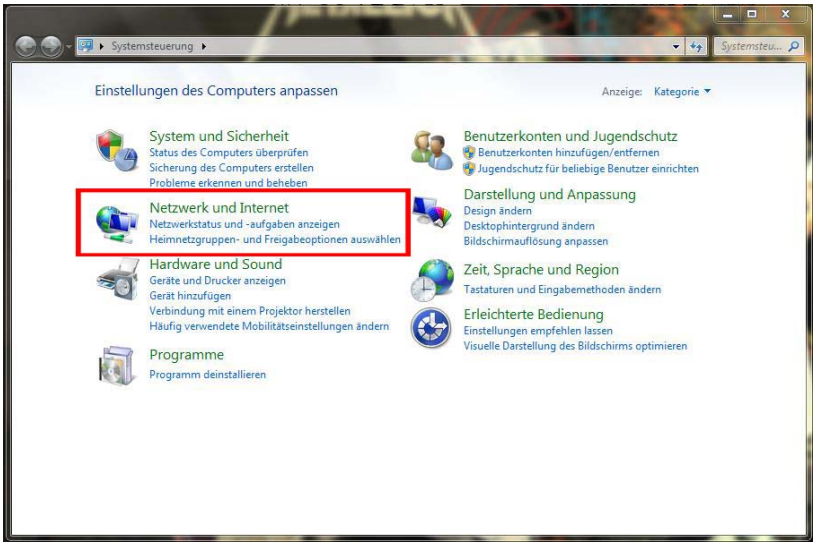


Abbildung 61

Anschließend muss man in das „Netzwerk- und Freigabecenter“ wechseln.

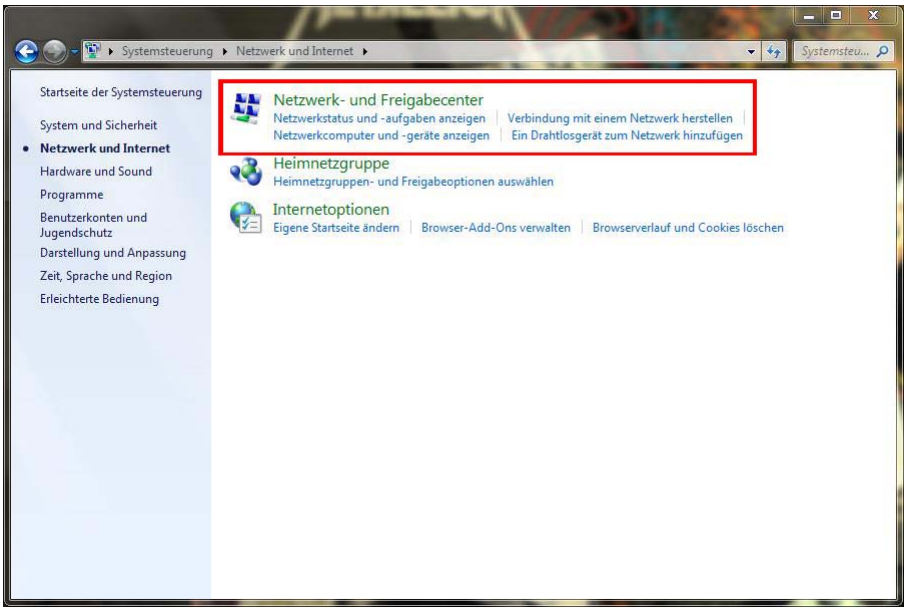


Abbildung 62

Im „Netzwerk- und Freigabecenter“ wählt man den Menüpunkt „Neue Verbindung oder neues Netzwerk einrichten“.

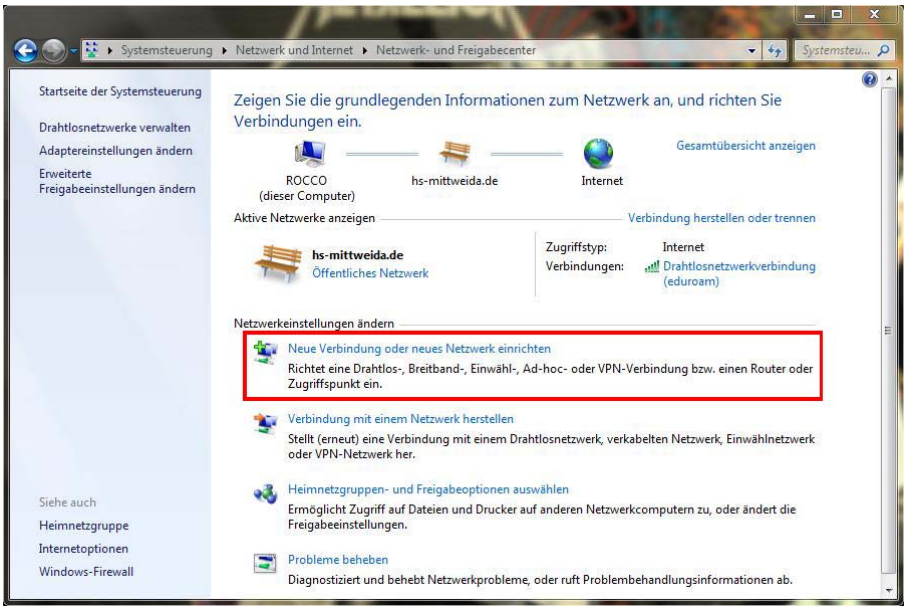


Abbildung 63

Danach öffnet sich ein neues Fenster mit einer großen Auswahl an Optionen. Ganz unten in der Liste befindet sich der Menüpunkt „ein drahtloses Ad-hoc-Netzwerk (Computer-zu-Computer) einrichten“, welches ausgewählt werden muss um mit der Einrichtung des Netzwerkes fortzufahren.

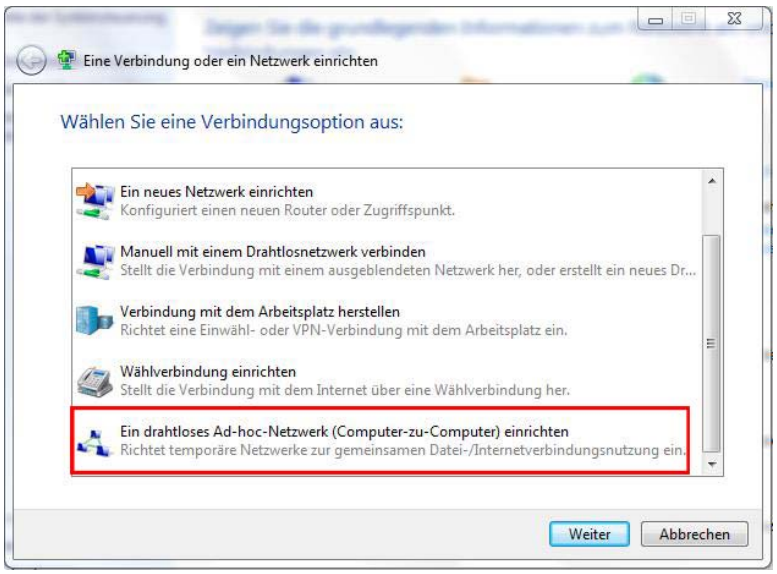


Abbildung 64

Nach dem Betätigen der „Weiter“ Taste öffnet sich ebenfalls wieder ein neues Fenster, auf dem man auch mittels „Weiter“ fortfahren muss.

Abbildung 65

Danach gelangt man zu einem kurzen Formular, welches mit dem Netzwerknamen, Sicherheitstyp und Sicherheitsschlüssel auszufüllen ist. Als Name und Schlüssel können beliebige Bezeichnungen vergeben werden, aber der Sicherheitstyp sollte „WEP“ sein.

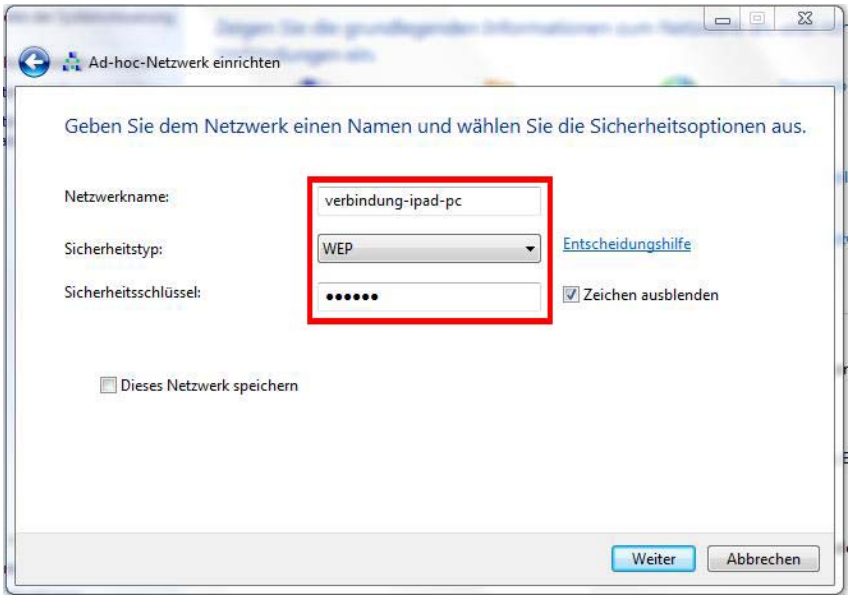


Abbildung 66

Nach der Bestätigung der Daten durch „Weiter“ erscheint eine Meldung mit dem Hinweis, dass das Netzwerk nun einsatzbereit ist.

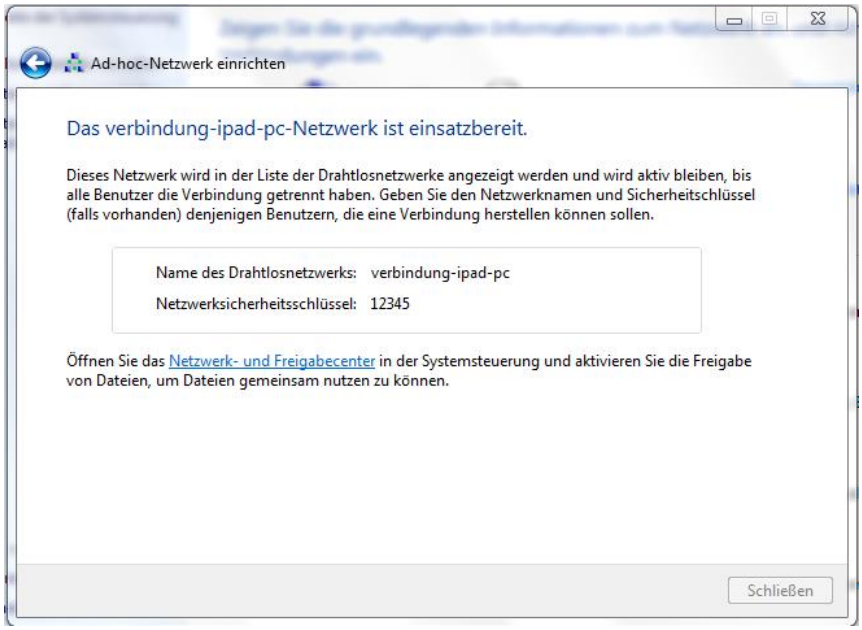


Abbildung 67

Auf dem iPad sollte das Netzwerk nun unter Einstellungen – WLAN aufgelistet werden. Bei der Verbindung wird nach dem Schlüssel gefragt, der gerade angelegt wurde. Nach Eingabe dieses Schlüssels verbinden sich die beiden Geräte.

Benutzerhandbuch des Webservers

Installation des Webservers (XAMPP)

Bevor man den Webserver benutzen kann, muss er installiert und konfiguriert werden. Die Installation von XAMPP ist relativ simple und in wenigen Schritten ausführbar. Zu Beginn kann die Setup-Datei kostenfrei auf apachefriends.org heruntergeladen werden. Nach dem Ausführen der Datei öffnet sich das Setup.

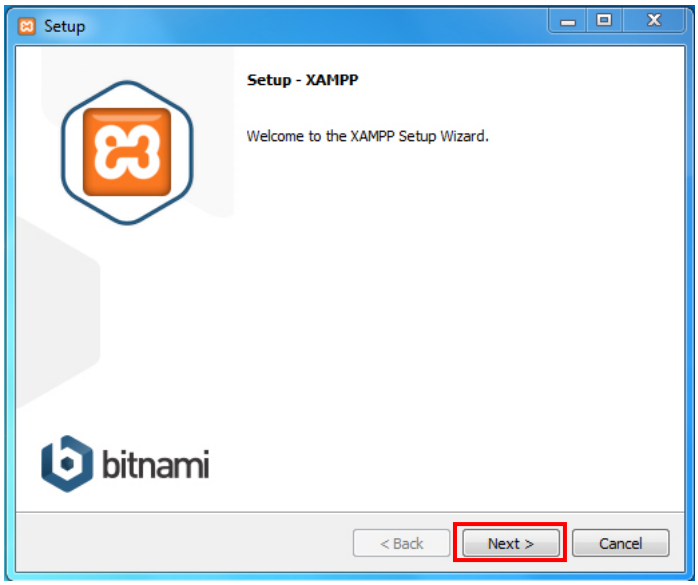


Abbildung 68

Um die Installation fortzusetzen muss die „next“ Taste betätigt werden.

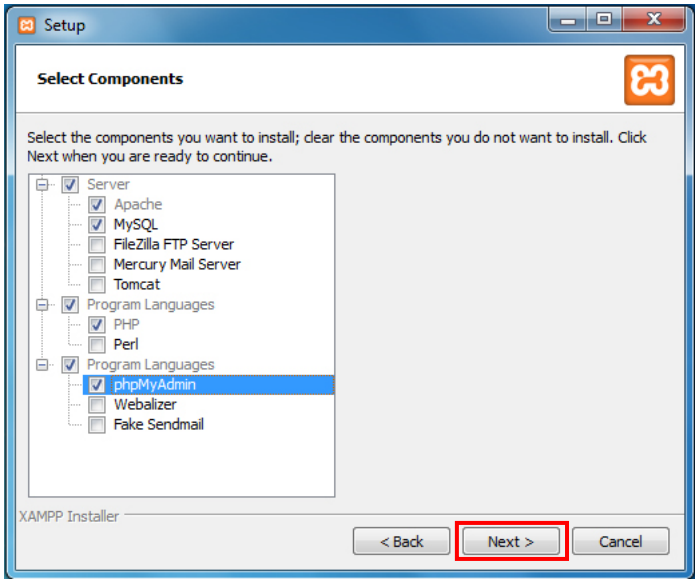


Abbildung 69

Danach wird festgelegt, welche Komponenten von XAMPP installiert werden sollen. Für den einfachen Datenaustausch zwischen PC und iPad genügt die Installation der Komponenten „Apache“, „MySQL“, „PHP“ und „phpMyAdmin“. Um fortzufahren, muss man mit „next“ bestätigen.

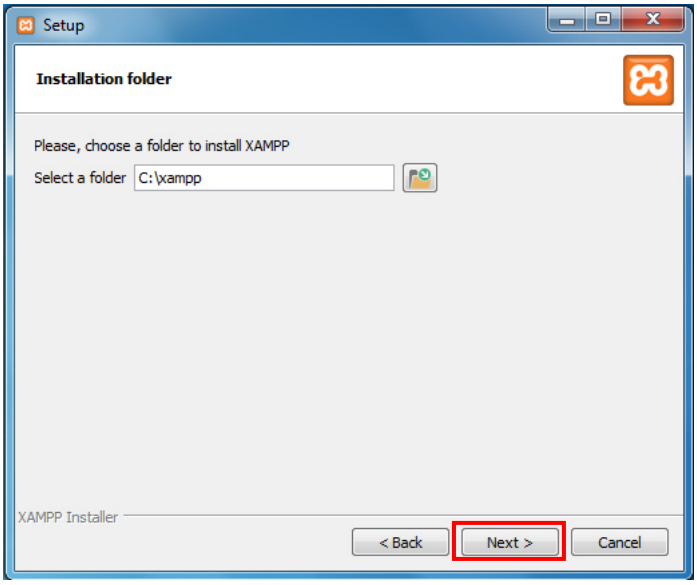


Abbildung 70

Nun muss noch der Speicherort des Webservers festgelegt werden. Als Standardkonfiguration bietet das Setup den Ordner „xampp“ auf „C“ an. Dieser Speicherort kann auch direkt verwendet werden.

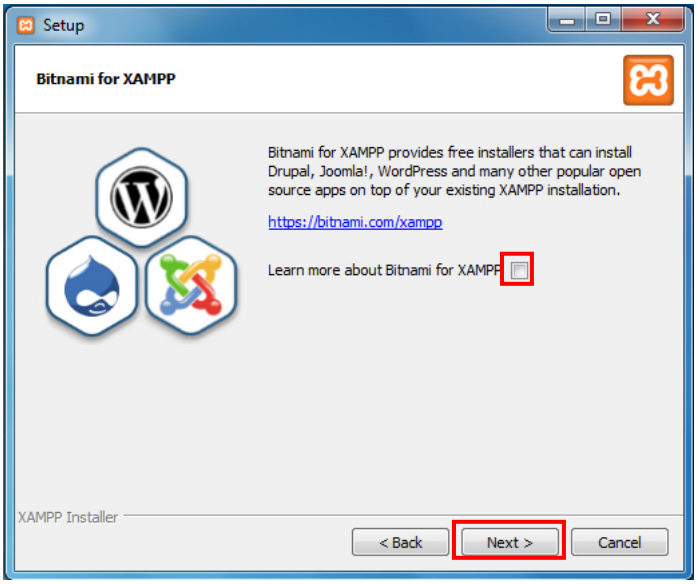


Abbildung 71

Im folgenden Fenster wird eine Information zu Bitnami angeboten, welche allerdings nicht unbedingt benötigt wird und durch entfernen des Häkchen ungenutzt bleibt. Um fortzufahren muss man „next“ wählen.

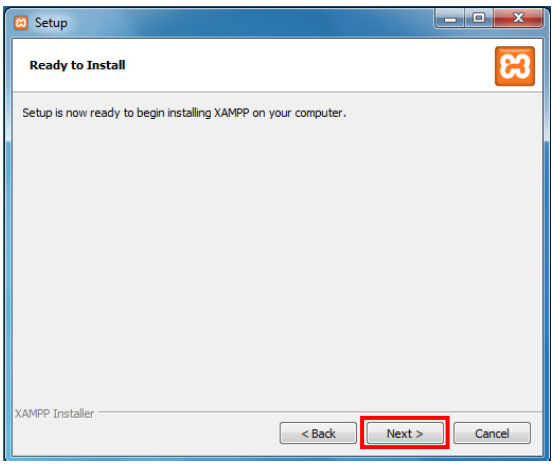


Abbildung 72

Um nun die Installation zu starten, muss nochmals die „next“-Taste betätigt werden.

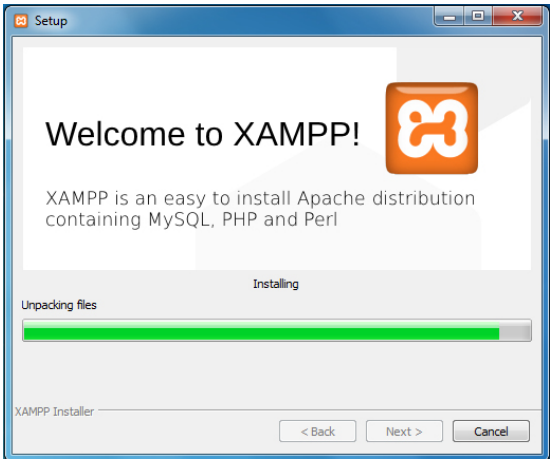


Abbildung 73

Danach öffnet sich ein Fenster mit einem Fortschrittsbalken, welcher anzeigt wie weit die Installation abgeschlossen ist. Ist dieser Balken vollständig wird der „next“ Button aktiviert. Durch Drücken dieses Buttons wird die Installation fortgesetzt.

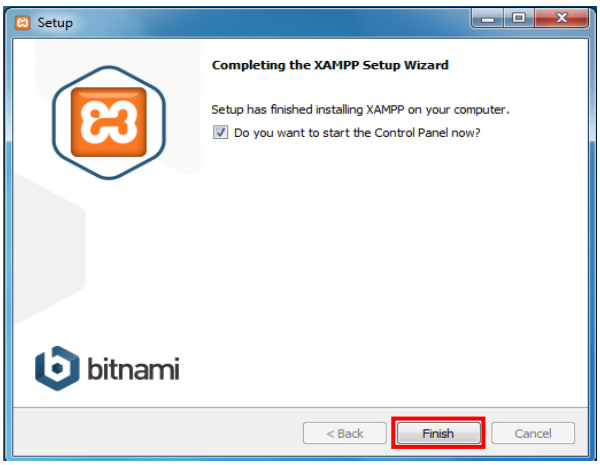


Abbildung 74

Nun ist die Installation abgeschlossen und man kann das Programm automatisch starten lassen. Dazu muss man nur noch den „finish“ Button wählen.

Abbildung 75

Beim Start des Programms, wird man nach der Standardsprache gefragt. Nach der jeweiligen Auswahl fährt man mit Drücken des „Speichern“ Buttons fort

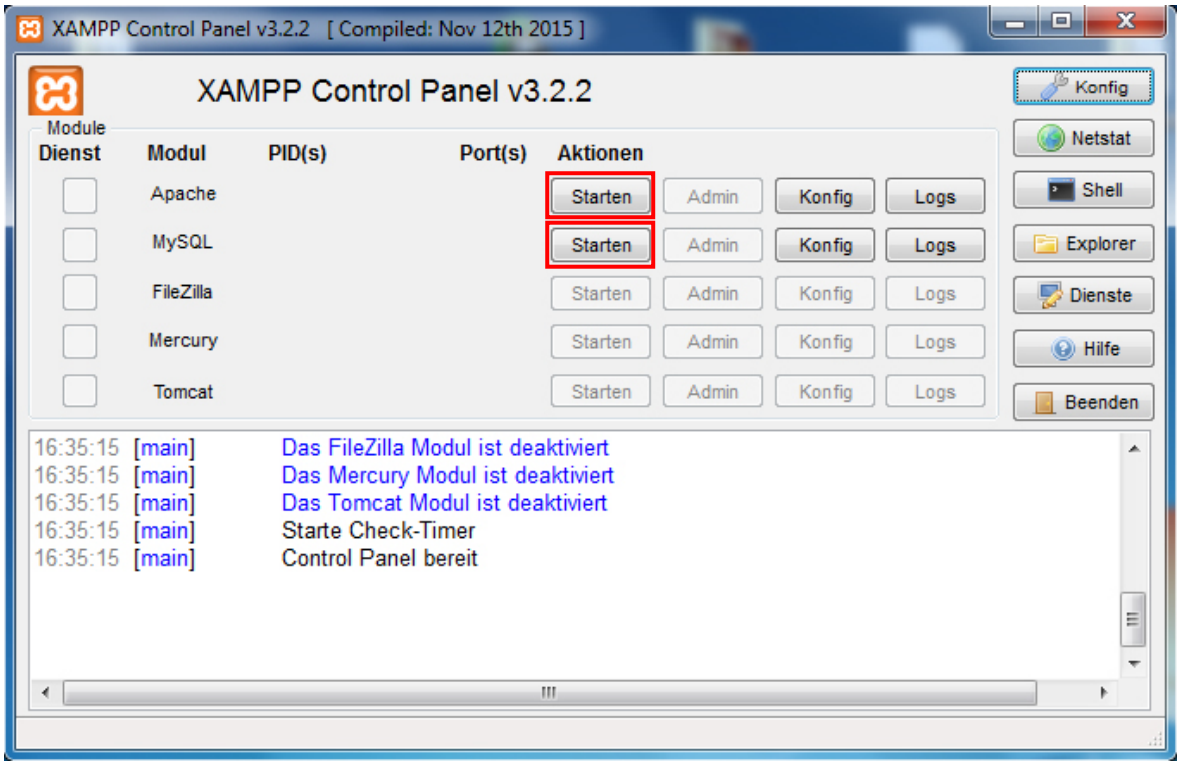


Abbildung 76

Jetzt öffnet sich das XAMPP Control Panel. Bevor der Webserver allerdings verwendet werden kann, müssen die Komponenten „Apache“ und „MySQL“ gestartet werden. Nun sind die Webseiten des Servers über den Internet-Browser erreichbar.

Konfiguration des Webservers (XAMPP)

Zum Einrichten des Webservers muss zuerst eine neue Datenbank mit phpMyAdmin angelegt werden. Dazu muss, nachdem in XAMPP Apache und MySQL gestartet wurden, ein Webbrowser aufgerufen werden und die Adresse „localhost/phpmyadmin“ in die Adresszeile geschrieben werden.

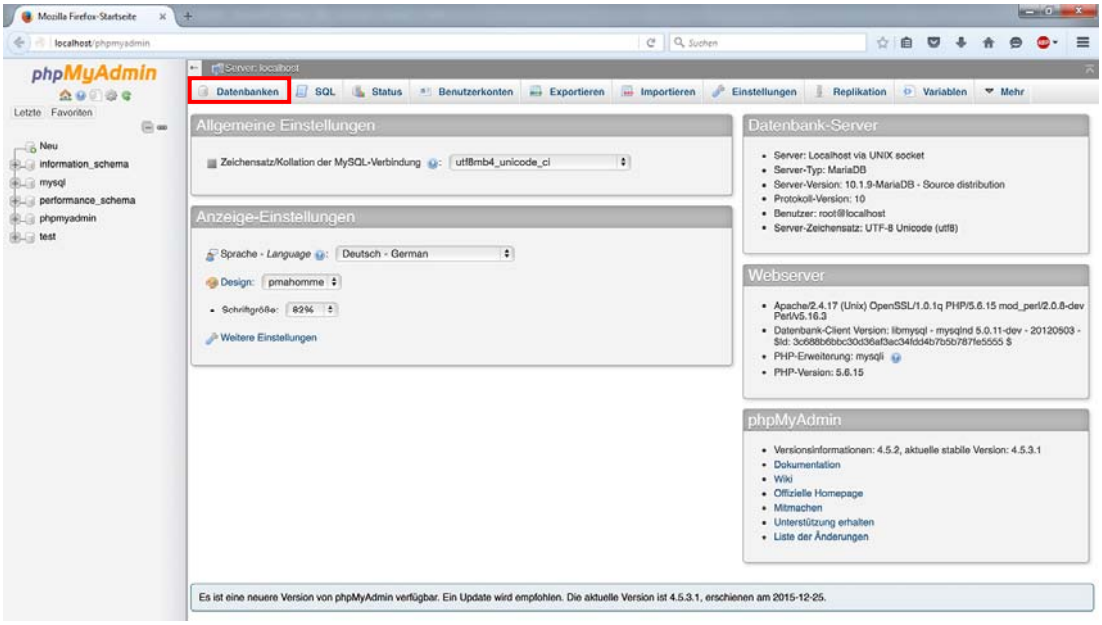


Abbildung 77

Nachdem die Startseite von phpMyAdmin aufgerufen wurde, muss per Klick auf den Menüpunkt „Datenbanken“ die Seite gewechselt werden.

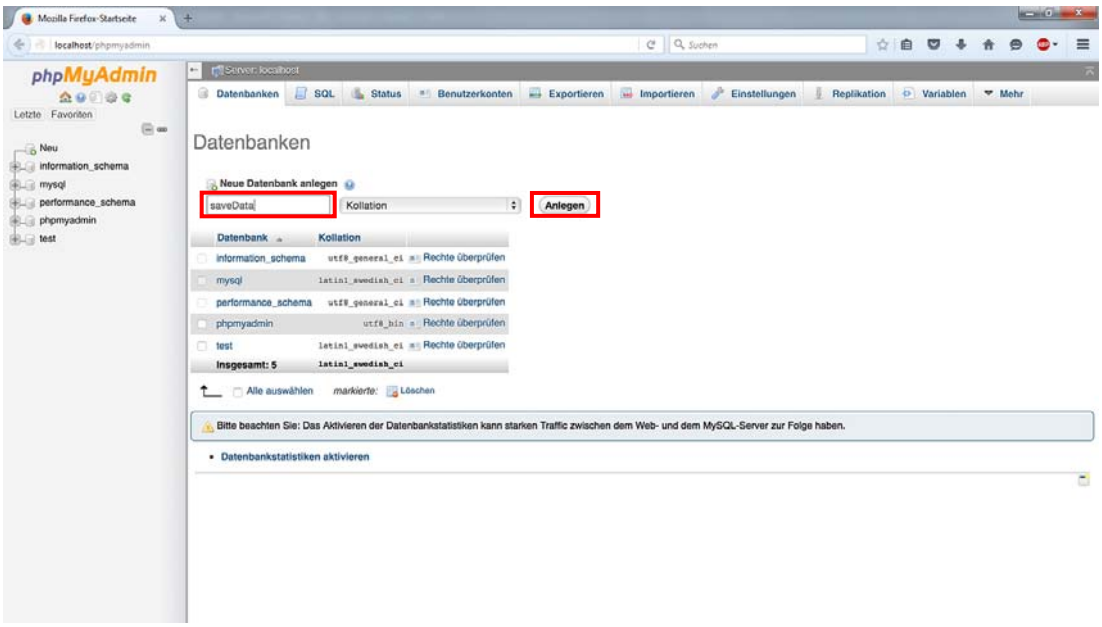


Abbildung 78

Um die Datenbank jetzt zu erstellen, muss der Name „saveData“ in das Textfeld eingegeben und mit „Anlegen“ bestätigt werden.

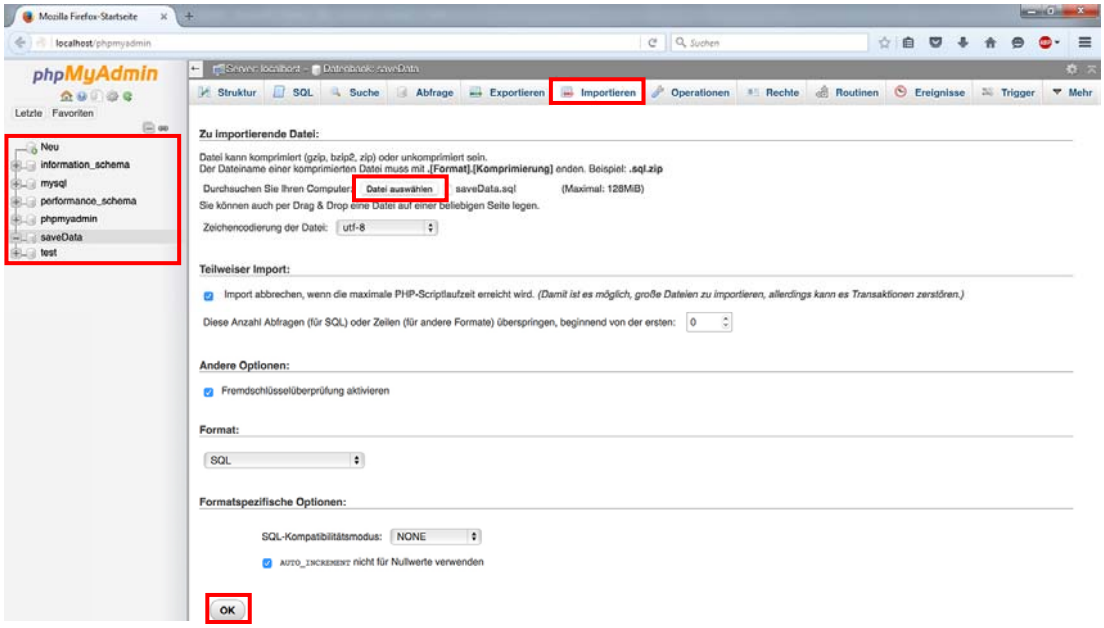


Abbildung 79

Man erkennt an der Auflistung der Datenbanken, ob die neue Datenbank erzeugt wurde. Durch das Anklicken von „saveData“ wird diese Datenbank ausgewählt und ist grau hinterlegt. Als nächstes muss der Menüpunkt Importieren ausgewählt werden. In dem mitgelieferten Ordner „FitnessVoter_ServerData“ befindet sich eine .sql Datei, welche hier importiert werden muss. Dazu wird einfach der Button „Datei auswählen“ betätigt und diese .sql Datei eingefügt. Danach muss nur noch mit „OK“ bestätigt werden und die Tabellen der Datenbank sind integriert.

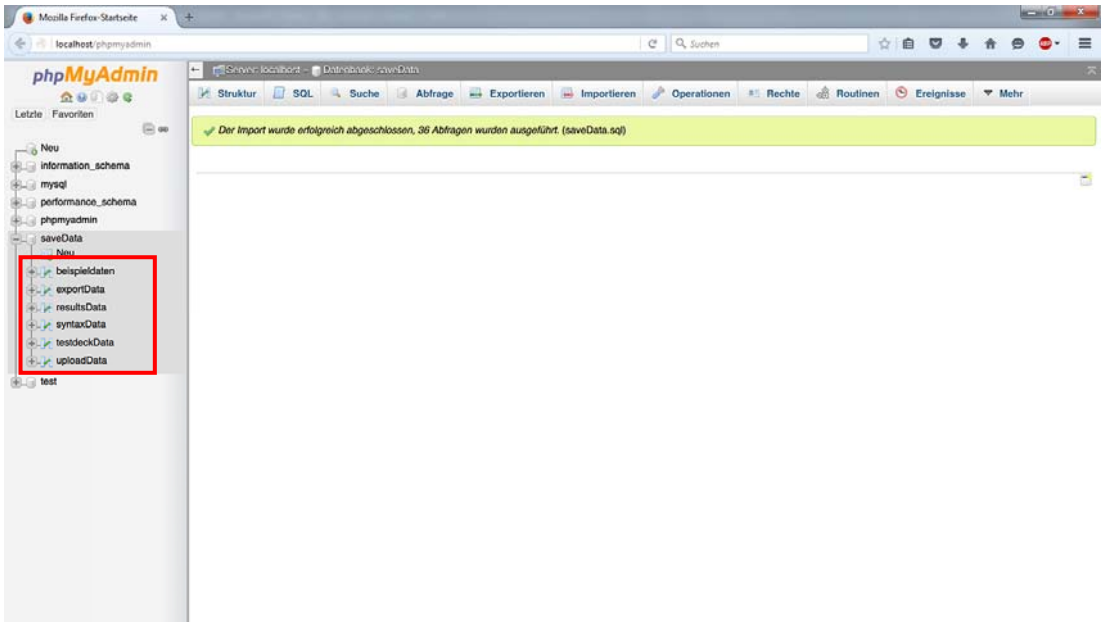


Abbildung 80

Mit Ausgabe der Meldung „Der Import wurde erfolgreich abgeschlossen.“ Ist der Import beendet. Die benötigten Tabellen der Datenbank sind nun links in der Liste zusehen.

Nun müssen nur noch die restlichen Dateien aus dem Ordner „FitnessVoter_ServerData“ in den htdocs-Ordner des Webservers kopiert werden. Dazu werden einfach alle Dateien des Ordners kopiert und unter dem folgenden Pfad eingefügt.

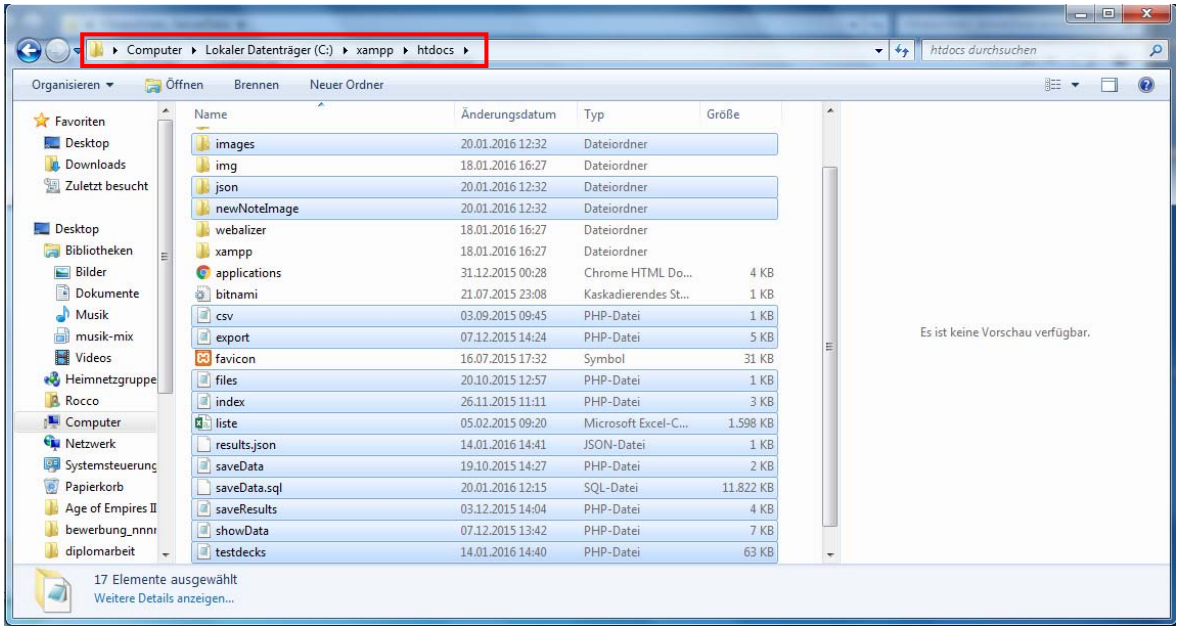


Abbildung 81

Unter Computer und dem Laufwerk C befindet sich ein Ordner mit dem Namen „xampp“. Dieser enthält den Ordner „htdocs“, welcher alle Dateien für die lokalen Webseiten beinhaltet. Darin werden die Dateien aus dem mitgelieferten Ordner eingefügt und können somit im Webbrowser aufgerufen werden.

Webserver Frontend

Nach dem der Webserver konfiguriert wurde, können die Webseiten im Webbrowser aufgerufen werden. Um auf die Startseite des FitnessVoter-Projektes zu gelangen, muss lediglich „localhost“ in die Adresszeile des Webbrowsers eingetragen werden.

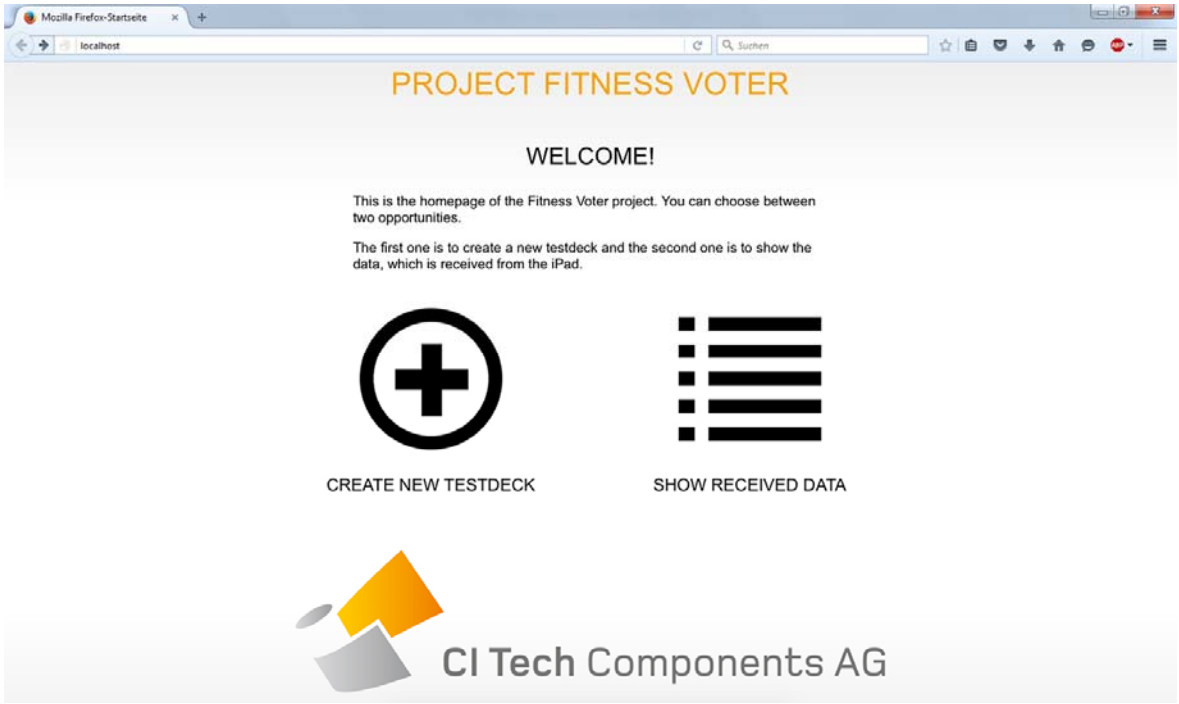


Abbildung 82

Auf der Startseite sind auch direkt die beiden möglichen Aktionen dargestellt. Der Administrator kann entweder mit „create new testedeck“ ein neues Testdeck erzeugen oder sich mittels „show received data“ die hochgeladenen Testdecks anzeigen lassen.

Beim Erzeugen eines neuen Testdecks muss ein Formular ausgefüllt werden.

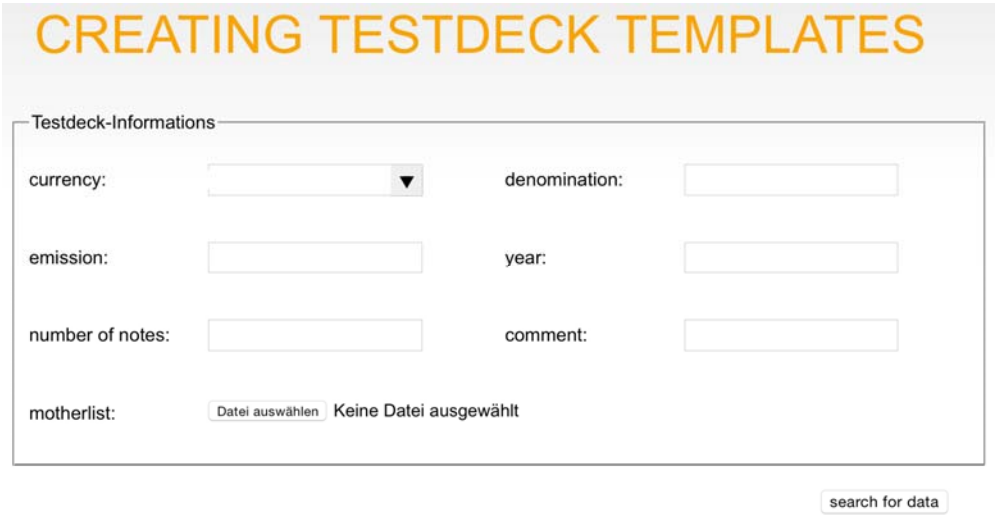


Abbildung 83

Zu Beginn müssen die Testdeckinformationen angegeben werden. Bei der Währung kann man mittels einer DropDown-Liste wählen.

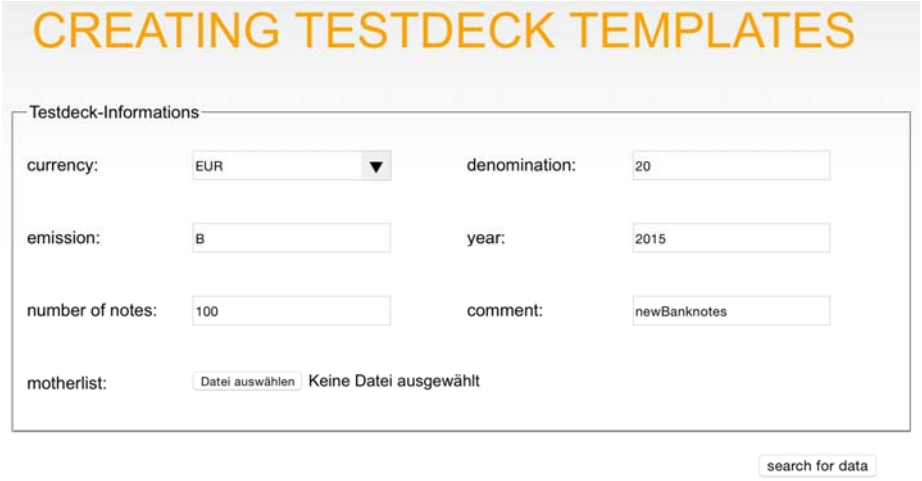


Abbildung 84

Nach dem Ausfüllen der Felder muss man den „search for data“ Button betätigen. Sind zu den eingegebenen Informationen bereits Daten vorhanden, so werden diese im Folgenden angezeigt.

Ist dies nicht der Fall, müssen diese Daten in den OCR-Informationen per Hand eingegeben werden.

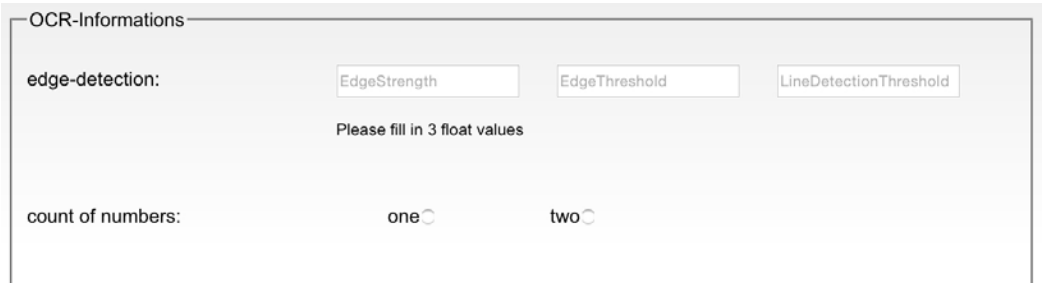


Abbildung 85

Zuerst müssen die Parameter für die Kantendetektion übergeben werden. Dabei muss der Schwellwert für den Filter „EdgeStrength“, die Kantenschwelle „EdgeThreshold“ und ein Wert für die Linienermittlung „LineDetectionThreshold“ angegeben werden.

Diese Werte sind mittels der Zusatzfunktion „Nr-Pos. Setup“ im Administrationsmenü der App testbar.

Danach ist die Anzahl der Seriennummern auf der Banknote anzugeben.

Als nächstes folgt die Eingabe der Seriennummer spezifischen Parameter. Enthält die Banknote zwei Seriennummern, müssen für beide die folgenden Werte angegeben werden.

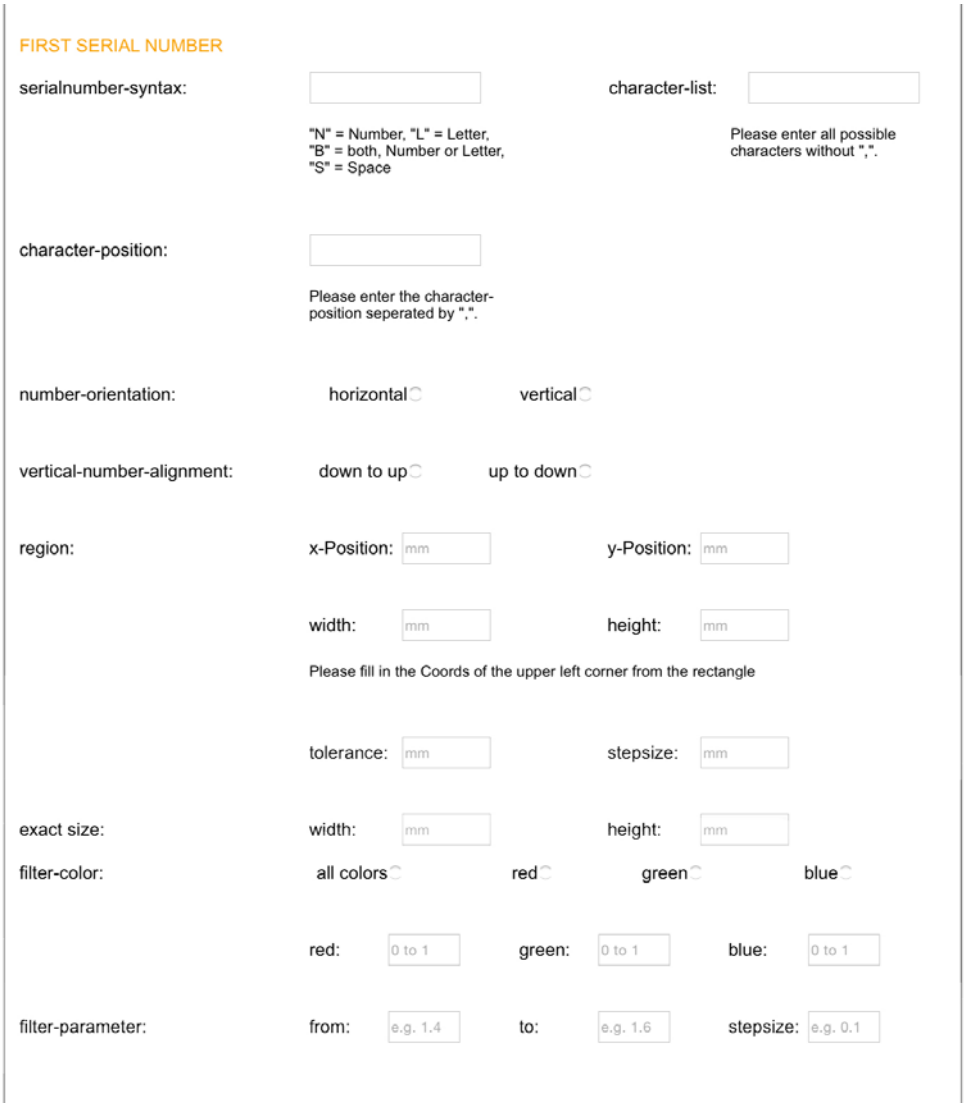
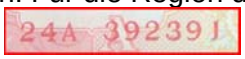


Abbildung 86

Für jede zu ermittelnde Seriennummer muss eine Syntax angegeben werden, welche aus Zahlen (N), Buchstaben (L), Beiden (B) oder Leerzeichen (S) bestehen kann. Am Beispiel der Seriennummer „AB12345“ würde die Syntax „LLNNNNN“ entsprechen. Danach wird eine Charakter-Liste angegeben, die alle Zeichen enthält, welche in der Seriennummer vorkommen können. Außerdem werden die Positionen der einzelnen Zeichen der Seriennummer angegeben, da es bei zwei vorhandenen Seriennummern vorkommen kann, dass eine in der anderen enthalten ist.

Ein weiterer wichtiger Parameter zur Seriennummer ist die Nummern-Lage. Wenn die Seriennummer senkrecht auf dem Geldschein gedruckt ist, muss noch eine Auswahl getroffen werden, ob die Nummer von oben nach unten ABC gedruckt wurde oder von unten nach oben ABC.

Als nächstes steht die Angabe der Region der Seriennummer an. Für die Region der Nummer können die Werte etwas großzügig gewählt werden. 

Dabei ist es immer wichtig von der oberen linken Ecke des Geldscheines auszugehen und die Werte in Millimetern anzugeben.

Außerdem wird eine Toleranz angegeben, mit der das Auslesen der Seriennummer erweitert werden kann, wenn keine Nummer in der Region gefunden wurde. Dazu muss auch eine Schrittgröße angegeben werden, die vorgibt um wieviel Millimeter die Region verschoben werden soll.

Aber es wird auch die exakte Größe  der Seriennummer benötigt, welche auch als Millimeter-Werte angegeben werden müssen.

Die letzten benötigten Parameter, sind mittels der Zusatzfunktion „filter-setup“ im Administartionsmenü der Applikation ablesbar. Zum einen wird die Farbe des Farbfilters benötigt und zum anderen die RGB-Werte für das Filtern des Bildes. Wird eine bestimmte Farbe, also Rot, Grün oder Blau als Farbe des Filters ausgewählt, so muss nur der jeweilige Wert für diese Farbe angegeben werden, da die anderen R-, G- oder B-Werte auf 0 gesetzt werden.

Die Werte der Filterparameter werden ebenfalls als float-Werte angegeben. Dabei muss ein Minimalwert und ein Maximalwert für den Filterparameter angegeben werden. Zusätzlich dazu wird noch eine Schrittgröße benötigt, die festlegt mit welchem Abstand die Parameter durchlaufen werden.

Mit dem Klick auf „create Testdeck“ wird das Testdeck erstellt und steht dann in der App zum Download bereit.

Auf der Seite „show received data“ können die hochgeladenen Testdecks gesichtet werden.

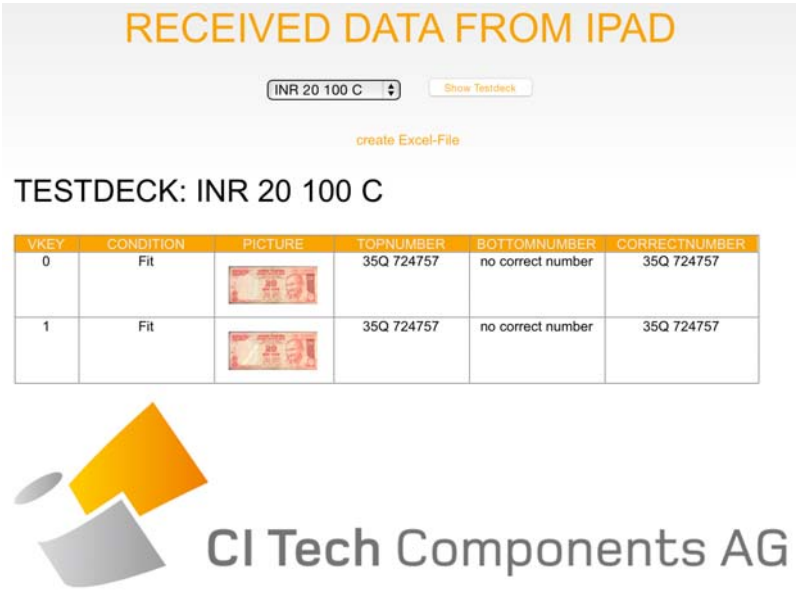


Abbildung 87

Das jeweilige Testdeck kann in einer DropDown-Liste ausgewählt werden und mit dem „show testdeck“ Button bestätigt werden. Danach werden die Daten des Testdecks in einer Tabelle dargestellt. Mittels des „create Excel-File“ Buttons, kann ein .xls File mit den Daten des Testdecks erstellt werden.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 05.02.2016

Rocco Oehme